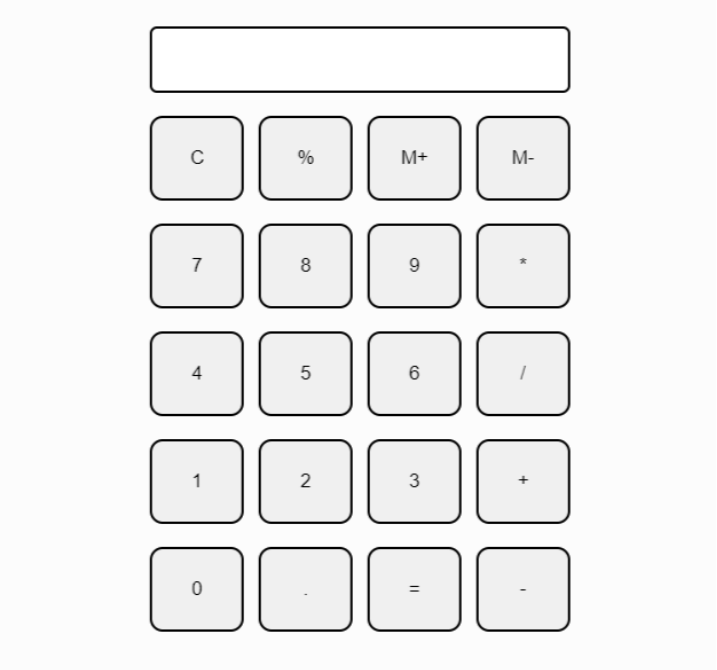
Hands – On Lab

Workshop 4

SIMPLE CALCULATOR

Create a UI of calculator using HTML and CSS and perform addition, subtraction, multiplication and division operations. Also handle the errors and exceptions. While clicking on C button, it should clear the textbox.




```
<html>
<head>
<style>

    .container{
       margin-left: 30px;
       margin-top: 60px;
       font-size: 20px;
    }
    div{
     padding: 3px 6px;
    }

    .display{
        width: 30px;
    }
</style>
```

```
</head>
<body>


<div class="container">
        <div class="display">
            <input id="display" name="display" disabled>
        </div>

        <div>
                <button id="button1" value="1">1</button>
            <button id="button2" value="2">2</button>
         <button id="button3" value="3">3</button>
        <button id="addButton" value="+">+</button>
        </div>

        <div>

            <button id="button4" value="4">4</button>
              <button id="button5" value="5">5</button>
              <button id="button6" value="6">6</button>
              <button id="subtractButton"value="-">-</button>

        </div>

        <div>
                <button id="button7" value="7">7</button>
                <button id="button8" value="8">8</button>
                <button id="button9" value="9">9</button>
                <button id="multiplyButton" value="*">*</button>
        </div>

        <div>
                <button id="clearButton"value="''">C</button>
              <button id="button0" value="0">0</button>
              <button id="equalsButton" value="=">=</button>
              <button id="divideButton" value="/">&#247;</button>
            </tr>
        </div>
    </div>
```

```
<script>
    const button9 = document.getElementById("button9");
    const button8 = document.getElementById("button8");
    const button7 = document.getElementById("button7");
    const button6 = document.getElementById("button6");
    const button5 = document.getElementById("button5");
    const button4 = document.getElementById("button4");
    const button3 = document.getElementById("button3");
    const button2 = document.getElementById("button2");
    const button1 = document.getElementById("button1");
    const button0 = document.getElementById("button0");
    const addButton = document.getElementById("addButton");
    const subtractButton = document.getElementById("subtractButton");
    const divideButton = document.getElementById("divideButton");
    const multiplyButton = document.getElementById("multiplyButton");
    const equalsButton = document.getElementById("equalsButton");
    const clearButton = document.getElementById("clearButton");

    const displaytext = document.getElementById('display')

    button9.addEventListener(('click'),()=>displaytext.value
+=button9.value)
    button8.addEventListener(('click'),()=>displaytext.value +=
button8.value)
    button7.addEventListener(('click'),()=>displaytext.value +=
button7.value)
    button6.addEventListener(('click'),()=>displaytext.value
+=button6.value)
    button5.addEventListener(('click'),()=>displaytext.value +=
button5.value)
    button4.addEventListener(('click'),()=>displaytext.value +=
button4.value)
    button3.addEventListener(('click'),()=>displaytext.value
+=button3.value)
    button2.addEventListener(('click'),()=>displaytext.value +=
button2.value)
    button1.addEventListener(('click'),()=>displaytext.value +=
button1.value)
    button0.addEventListener(('click'),()=>displaytext.value +=
button0.value)
    addButton.addEventListener(('click'),()=>displaytext.value +=
addButton.value)
```

```
    subtractButton.addEventListener(('click'),()=>displaytext.value +=
subtractButton.value)
    divideButton.addEventListener(('click'),()=>displaytext.value +=
divideButton.value)
    multiplyButton.addEventListener(('click'),()=>displaytext.value +=
multiplyButton.value)
    clearButton.addEventListener('click',()=>displaytext.value =
eval(clearButton.value))
    equalsButton.addEventListener('click',()=>displaytext.value =
eval(displaytext.value))
</script>


</body>



</html>
```

DIGITAL CLOCK

Create a Digital Clock using setInterval and Date function in JavaScript.

Note: Date object can be used to get the date, time, hours and seconds which can be updated using setInterval (). Try to keep the UI good looking and different from each other.

```
 <!DOCTYPE html>
    <html lang="en">
      <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>JavaScript Digital Clock</title>
```

```
        <link rel="stylesheet" href="style.css">
        <script src="main.js"></script>
      </head>
      <style>
      .container{
        text-align:center ;
        display: flex;

       border-radius: 10px;
        border: 2px solid black;
       background: black;
       color: white;
       width: 250px;
       height: 50px;
      font-size: 40px;

     margin: 153px 200px;
      }
    div{
      padding: 3px 6px;
    }

    </style>
<body>
    <div class="container">

        <div id="hours" >00</div>
        <div>:</div>
        <div id="minutes">00</div>
        <div>:</div>
        <div id="seconds">00</div>
        <div id="session">AM</div>
    </div>
</body>
</html>
```

```
function displayTime(){
    var dateTime = new Date();
    var hrs = dateTime.getHours();
    var min = dateTime.getMinutes();
    var sec = dateTime.getSeconds();
    var session = document.getElementById('session');
```

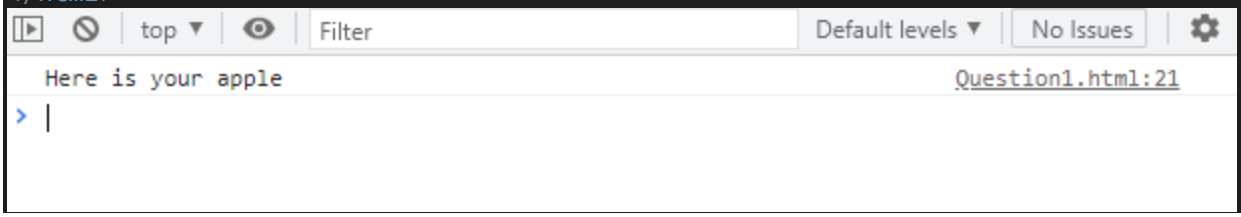
```
    if(hrs >= 12){
        session.innerHTML = 'PM';
    }else{
        session.innerHTML = 'AM';
    }

    if(hrs > 12){
        hrs = hrs - 12;
    }

    document.getElementById('hours').innerHTML = hrs;
    document.getElementById('minutes').innerHTML = min;
    document.getElementById('seconds').innerHTML = sec;
}
setInterval(displayTime, 10);
```

ASYNCHRONOUS JAVASCRIPT

a. Create a function called **getFruit** that takes in a fruit name as a parameter and returns a Promise that resolves after 1 second with a message saying "Here is your [fruit]". If the fruit name is "watermelon", the Promise should reject after 2 seconds with an error message saying "Sorry, we're out of watermelons

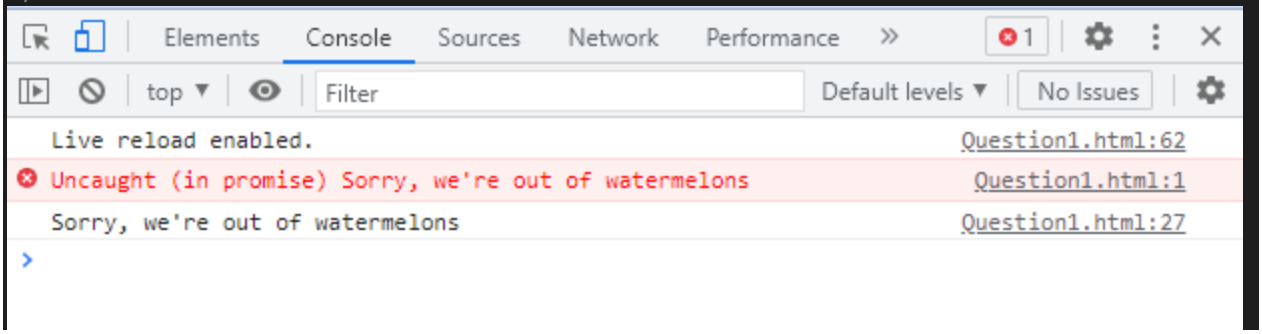
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Question1</title>
</head>
<body>

    <script>
```

```
        function Question1(pair){
        const Question1 = new Promise(function(resolve,reject) {
if (pair == "watermelon") {
reject ("Sorry, we're out of watermelons");
}
     resolve(`Here is your ${pair}`);
});
Question1.then(function(outcome){
    setTimeout(function (){
        console.log(outcome);
    },1000);
});

Question1.catch(function (outcome) {
   setTimeout(function () {
    console.log(outcome);
   } ,2000);
});
}
Question1("apple")

    </script>
</body>
</html>
```




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Question1</title>
</head>
<body>

    <script>
        function Question1(pair){
        const Question1 = new Promise(function(resolve,reject) {
if (pair == "watermelon") {
```

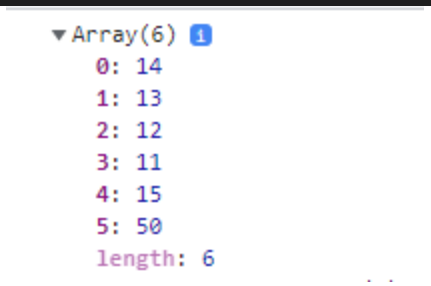
```
reject ("Sorry, we're out of watermelons");
}
     resolve(`Here is your ${pair}`);
});
Question1.then(function(outcome){
    setTimeout(function (){
        console.log(outcome);
    },1000);
});

Question1.catch(function (outcome) {
   setTimeout(function () {
    console.log(outcome);
   } ,2000);
});
}
Question1("watermelon")

    </script>
</body>
</html>
```



b. Create a function called **arrayManipulation** that takes in an array of numbers and two callback functions as parameters. The first callback function should perform an operation on each element in the array, and the second callback function should filter the resulting array based on a condition. The function should return the filtered array

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Question2</title>
```

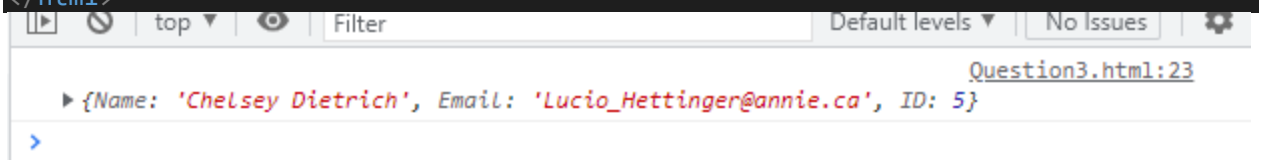
```
</head>
<body>
    <script>
        function Question2(filter){
            return filter + 5;
        }

        function Question1(filter){
            return filter > 10;
        }

        function Niraj(ab, cd , ef){
            let num1 = ab.map(cd);
            let num2 = num1.filter(ef);
            return num2;
        }
let ab = [9,8,7,6,5,4,3,2,10,,1,45];
let Fixed = Niraj(ab,Question2,Question1);
console.log(Fixed);

    </script>
</body>
</html>
```



c. Create an asynchronous function called **fetchUserData** that uses **async/await** to fetch user data from a JSON API (**https://jsonplaceholder.typicode.com/users**). The function should take in a user ID as a parameter, and use that ID to fetch the user's data from the API. If the API returns an error, the function should throw an error. If the API returns the user data, the function should return an object containing the user's name and email

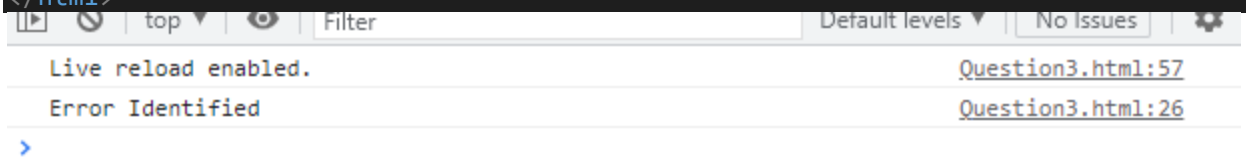
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Document</title>
</head>
<body>
    <script>
        async function question3(userID){
            let Niraj = await
fetch("https://jsonplaceholder.typicode.com/users");
            let data = await Niraj.json();
            return {
                Name: data[userID-1].name,
                Email: data[userID - 1].email,
                ID: data[userID-1].id,
            };

 }
 question3(5)
 .then(function(data){
console.log(data);
 })
 .catch(function(){
    console.log("Error Identified");
 });
    </script>
</body>
</html>
```




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Document</title>
</head>
<body>
    <script>
        async function question3(userID){
            let Niraj = await
fetch("https://jsonplaceholder.typicode.com/users");
            let data = await Niraj.json();
            return {
                Name: data[userID-1].name,
                Email: data[userID - 1].email,
                ID: data[userID-1].id,
            };

 }
 question3(5000)
 .then(function(data){
console.log(data);
 })
 .catch(function(){
    console.log("Error Identified");
 });
    </script>
</body>
</html>
```



d. Fetch data from API (https://jsonplaceholder.typicode.com/todos) and display (UserId,Title and Status) in a browser whose completed status is true and id<=50

```
const Niraj = document.querySelector(".demo");
  function question(){
    fetch("https://jsonplaceholder.typicode.com/todos")

      .then(function(response){
        console.log(response.status);
       return response.json();
      })
      .then(function(data){
          console.log(data);
```

```
          const filtered =data.filter(function(data){
            return data.id <= 50 && data.completed == true;

      });
      console.log(filtered);
      const userID = filtered.map(function(data){
          const ID = data.userId;
          const title =data.title;
          const dataStatus =data.completed;
          return `User ID:${ID} | Title: ${title} | Status:
${dataStatus}`;
      });

      Niraj.innerHTML= userID;

  });
  }
  question();
```

User ID:1 | Title: et porro tempora | Status: true,User ID:1 | Title: quo adipisci enim quam ut ab | Status: true,User ID:1 | Title: illo est ratione doloremque quia maiores aut | Status: true,User ID:1 | Title: vero rerum temporibus dolor | Status: true,User ID:1 | Title: ipsa repellendus fugit nisi | Status: true,User ID:1 | Title: repellendus sunt dolores architecto voluptatum | Status: true,User ID:1 | Title: ab voluptatum amet voluptas | Status: true,User ID:1 | Title: accusamus eos facilis sint et aut voluptatem | Status: true,User ID:1 | Title: quo laboriosam deleniti aut qui | Status: true,User ID:1 | Title: molestiae ipsa aut voluptatibus pariatur dolor nihil | Status: true,User ID:1 | Title: ullam nobis libero sapiente ad optio sint | Status: true,User ID:2 | Title: distinctio vitae autem nihil ut molestias quo | Status: true,User ID:2 | Title: voluptas quo tenetur perspiciatis explicabo natus | Status: true,User ID:2 | Title: aliquam aut quasi | Status: true,User ID:2 | Title: veritatis pariatur delectus | Status: true,User ID:2 | Title: nemo perspiciatis repellat ut dolor libero commodi blanditiis omnis | Status: true,User ID:2 | Title: repellendus veritatis molestias dicta incidunt | Status: true,User ID:2 | Title: excepturi deleniti adipisci voluptatem et neque optio illum ad | Status: true,User ID:2 | Title: totam atque quo nesciunt | Status: true,User ID:3 | Title: tempore ut sint quis recusandae | Status: true,User ID:3 | Title: cum debitis quis accusamus doloremque ipsa natus sapiente omnis | Status: true,User ID:3 | Title: cupiditate necessitatibus ullam aut quis dolor voluptate | Status: true