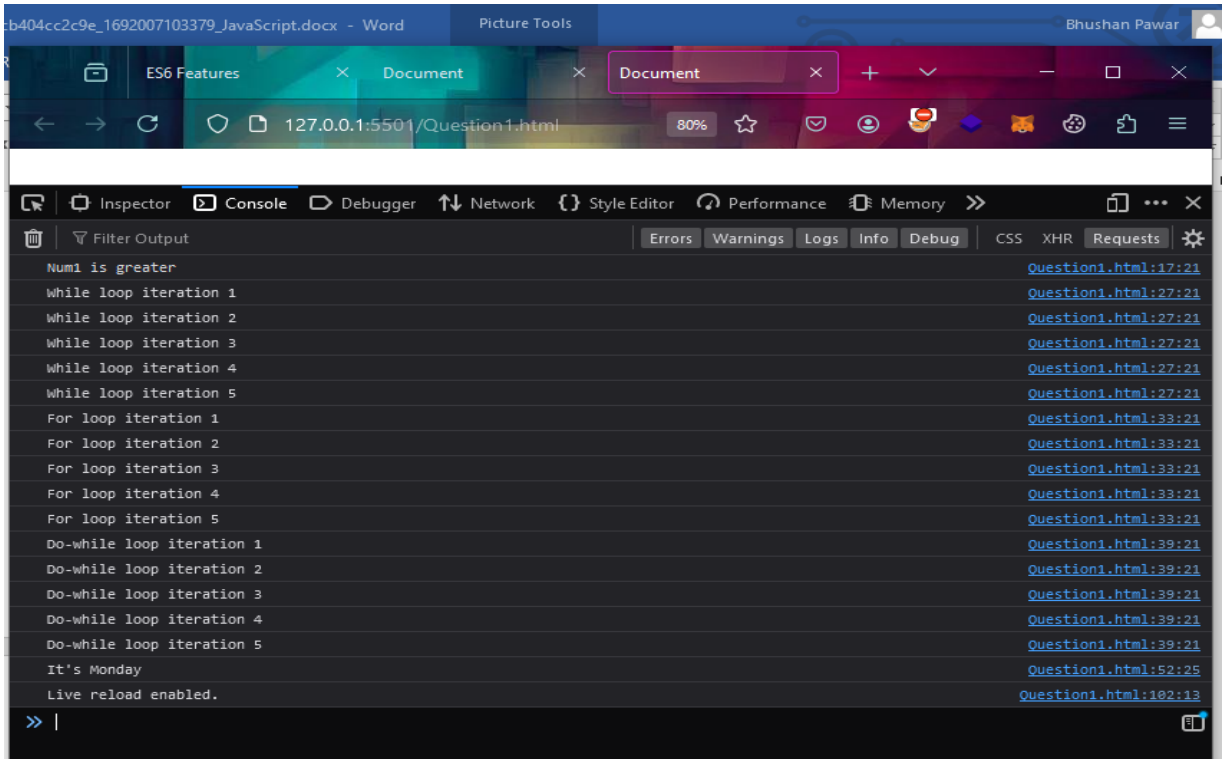
1. Write a program to demonstrate JavaScript loops, operators and conditions?

Code :

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

<body>
  <script>
    // Conditional statement to compare two numbers
    let num1 = 10;
    let num2 = 5;

    if (num1 > num2) {
      console.log("Num1 is greater");
    } else if (num1 < num2) {
      console.log("Num2 is greater");
    } else {
      console.log("Num1 and Num2 are equal");
    }

    // While loop
    let i = 1;
    while (i <= 5) {
      console.log("While loop iteration " + i);
      i++;
    }

    // For loop
    for (let j = 1; j <= 5; j++) {
      console.log("For loop iteration " + j);
    }

    // Do-while loop
    let k = 1;
    do {
      console.log("Do-while loop iteration " + k);
      k++;
    } while (k <= 5);
    let today = new Date();
    let dayOfWeek = today.getDay();

    switch (dayOfWeek) {
```

```
        case 0:
            console.log("It's Sunday");
            break;
        case 1:
            console.log("It's Monday");
            break;
        case 2:
            console.log("It's Tuesday");
            break;
        case 3:
            console.log("It's Wednesday");
            break;
        case 4:
            console.log("It's Thursday");
            break;
        case 5:
            console.log("It's Friday");
            break;
        case 6:
            console.log("It's Saturday");
            break;
        default:
            console.log("Unknown day");
    }
  </script>
</body>
</html>
```

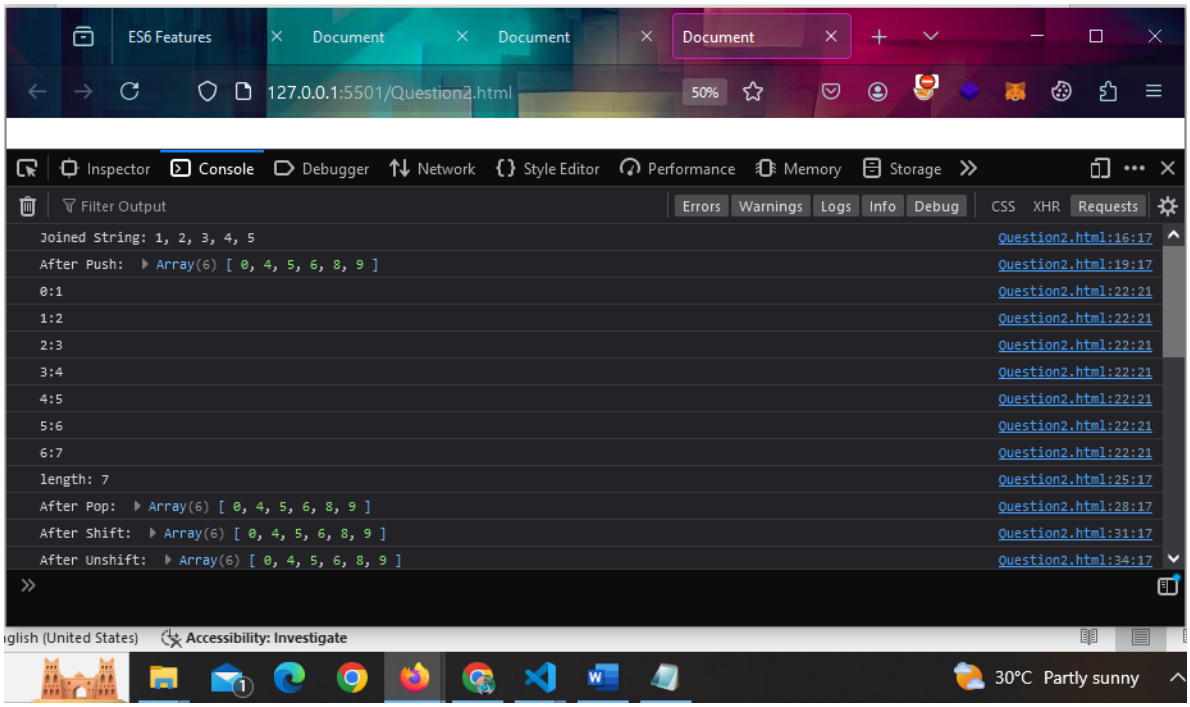


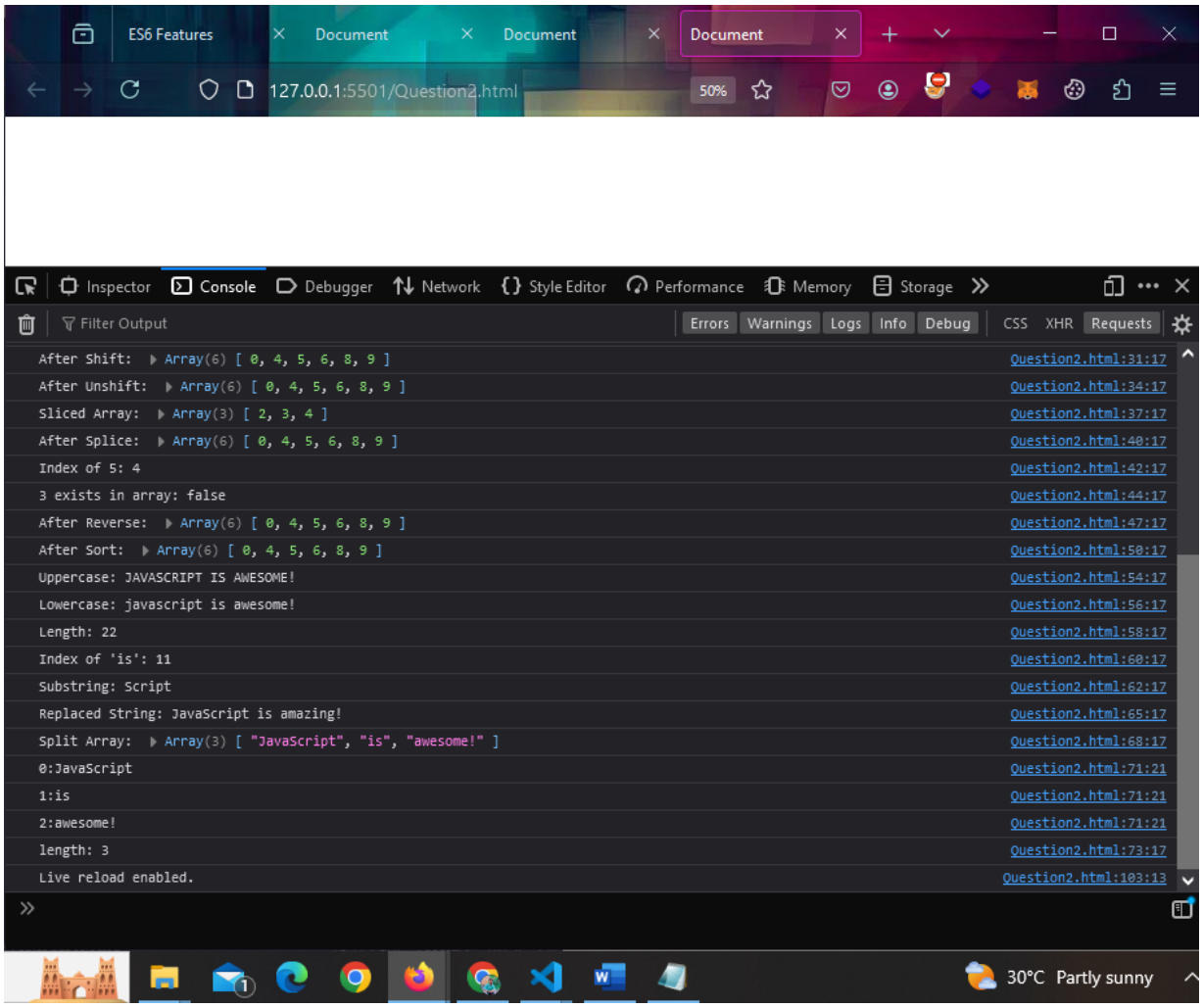
2. Write a program to demonstrate different array and string methods in JavaScript?

Code :

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

<body>
  <script>
    // Array methods demonstration
    let array = [1, 2, 3, 4, 5];

    let joinedArray = array.join(", ");
    console.log("Joined String: " + joinedArray);

    array.push(6, 7);
    console.log("After Push: ", array);

    let mappedArray = array.map((x, index) => {
      console.log(index + ":" + x);
      return x + 3;
    });
    console.log("length: " + mappedArray.length);

    array.pop();
    console.log("After Pop: ", array);

    array.shift();
    console.log("After Shift: ", array);

    array.unshift(0);
    console.log("After Unshift: ", array);

    let slicedArray = array.slice(1, 4);
    console.log("Sliced Array: ", slicedArray);

    array.splice(1, 2, 8, 9);
    console.log("After Splice: ", array);

    console.log("Index of 5: " + array.indexOf(5));

    console.log("3 exists in array: " + array.includes(3));

    array.reverse();
```

```
    console.log("After Reverse: ", array);

    array.sort((a, b) => a - b);
    console.log("After Sort: ", array);

    let str = "JavaScript is awesome!";

    console.log("Uppercase: " + str.toUpperCase());

    console.log("Lowercase: " + str.toLowerCase());

    console.log("Length: " + str.length);

    console.log("Index of 'is': " + str.indexOf('is'));

    console.log("Substring: " + str.substring(4, 10));

    let newStr = str.replace("awesome", "amazing");
    console.log("Replaced String: " + newStr);

    let splitStr = str.split(" ");
    console.log("Split Array: ", splitStr);

    splitStr.forEach((element, index) => {
      console.log(index + ":" + element);
    });
    console.log("length: " + splitStr.length);
  </script>
</body>

</html>
```

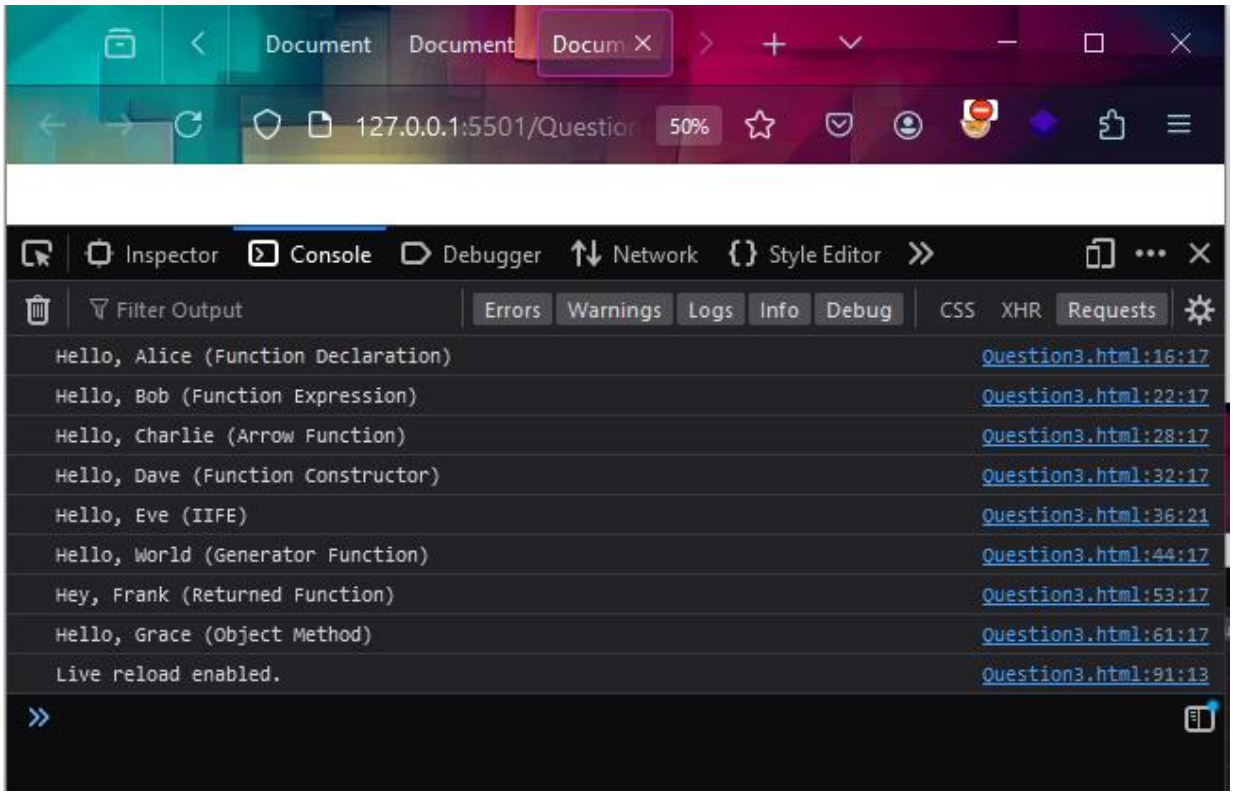



ES6 Features
Document
Document
Document
127.0.0.1:5501/Question2.html
50%
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
After Shift: Array(6) [0, 4, 5, 6, 8, 9] Question2.html:31:17
After Unshift: Array(6) [0, 4, 5, 6, 8, 9] Question2.html:34:17
Sliced Array: Array(3) [2, 3, 4] Question2.html:37:17
After Splice: Array(6) [0, 4, 5, 6, 8, 9] Question2.html:40:17
Index of 5: 4 Question2.html:42:17
3 exists in array: false Question2.html:44:17
After Reverse: Array(6) [0, 4, 5, 6, 8, 9] Question2.html:47:17
After Sort: Array(6) [0, 4, 5, 6, 8, 9] Question2.html:50:17
Uppercase: JAVASCRIPT IS AWESOME! Question2.html:54:17
Lowercase: javascript is awesome! Question2.html:56:17
Length: 22 Question2.html:58:17
Index of 'is': 11 Question2.html:60:17
Substring: Script Question2.html:62:17
Replaced String: JavaScript is amazing! Question2.html:65:17
Split Array: Array(3) ["JavaScript", "is", "awesome!"] Question2.html:68:17
0:JavaScript Question2.html:71:21
1:is Question2.html:71:21
2:awesome! Question2.html:71:21
length: 3 Question2.html:73:17
Live reload enabled. Question2.html:103:13
30°C Partly sunny

3. Write a program to show different ways to create a function in JavaScript?

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script>
    // Function Declaration
    function greetDeclaration(name) {
      return `Hello, ${name} (Function Declaration)`;
    }
    console.log(greetDeclaration("Alice"));

    // Function Expression
    const greetExpression = function(name) {
      return `Hello, ${name} (Function Expression)`;
    };
    console.log(greetExpression("Bob"));

    // Arrow Function
    const greetArrow = (name) => {
      return `Hello, ${name} (Arrow Function)`;
    };
    console.log(greetArrow("Charlie"));

    // Function Constructor
    const greetConstructor = new Function('name', 'return `Hello, ${name} (Function
Constructor)`;');
    console.log(greetConstructor("Dave"));

    // Immediately Invoked Function Expression (IIFE)
    (function(name) {
      console.log(`Hello, ${name} (IIFE)`);
    })("Eve");

    // Generator Function
    function* greetGenerator() {
      yield "Hello, World (Generator Function)";
    }
    const generator = greetGenerator();
    console.log(generator.next().value);

    // Function that returns another function
    function createGreeter() {
      return function(name) {
```

```
            return `Hey, ${name} (Returned Function)`;
        };
    }
    const returnedFunction = createGreeter();
    console.log(returnedFunction("Frank"));

    // Object Method
    const obj = {
        greet(name) {
            return `Hello, ${name} (Object Method)`;
        }
    };
    console.log(obj.greet("Grace"));
  </script>
</body>

</html>
```



4. Write a program to implement pomodoro using JavaScript DOM?

Code :

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Pomodoro Timer</title>

  <style>
    body {
      background-color: #000;
      color: #fff;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      height: 100vh;
      margin: 0;
      font-family: Arial, sans-serif;
    }

    .header {
      margin-bottom: 0px;
      font-size: 14px;
      text-align: center;
      color: #838181;
    }

    .labels {
      padding: 15px;
      display: flex;
      justify-content: space-around;
      width: 100%;
      background-color: #222121;
      font-family: Arial, Helvetica, sans-serif;
    }

    .label {
      margin-bottom: 10px;
      margin-top: 10px;
      font-size: 14px;
      text-align: center;
      color: rgb(126, 127, 127);
    }

    .timer {
      font-size: 10rem;
      margin-bottom: 20px;
```

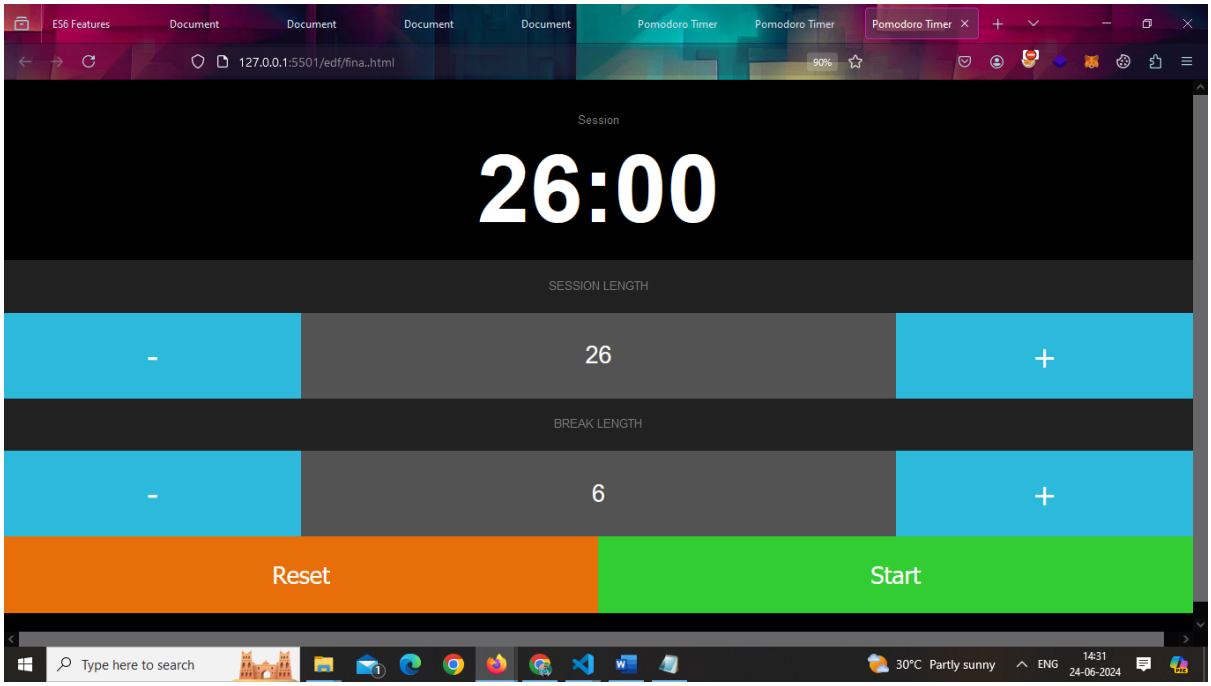
```
    margin-top: 10px;
    color: rgb(255, 255, 255);
}

.controls {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    justify-content: space-around;
    width: 100%;
}

.control1,
.control2 {
    flex: 1;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 30px;
    padding: 30px;
    color: white;
    cursor: pointer;
}

.control1 {
    background-color: rgb(83, 83, 83);
}

.control2 {
    background-color: rgb(44, 185, 220);
    border: none;
    font-size: 40px;
}

.buttons {
    display: flex;
    justify-content: space-between;
    width: 100%;
}

.buttons button {
    flex: 1;
    padding: 30px;
    font-size: 30px;
    border: none;
    color: white;
    cursor: pointer;
}

.reset {
    background-color: #e86e0a;
}
```

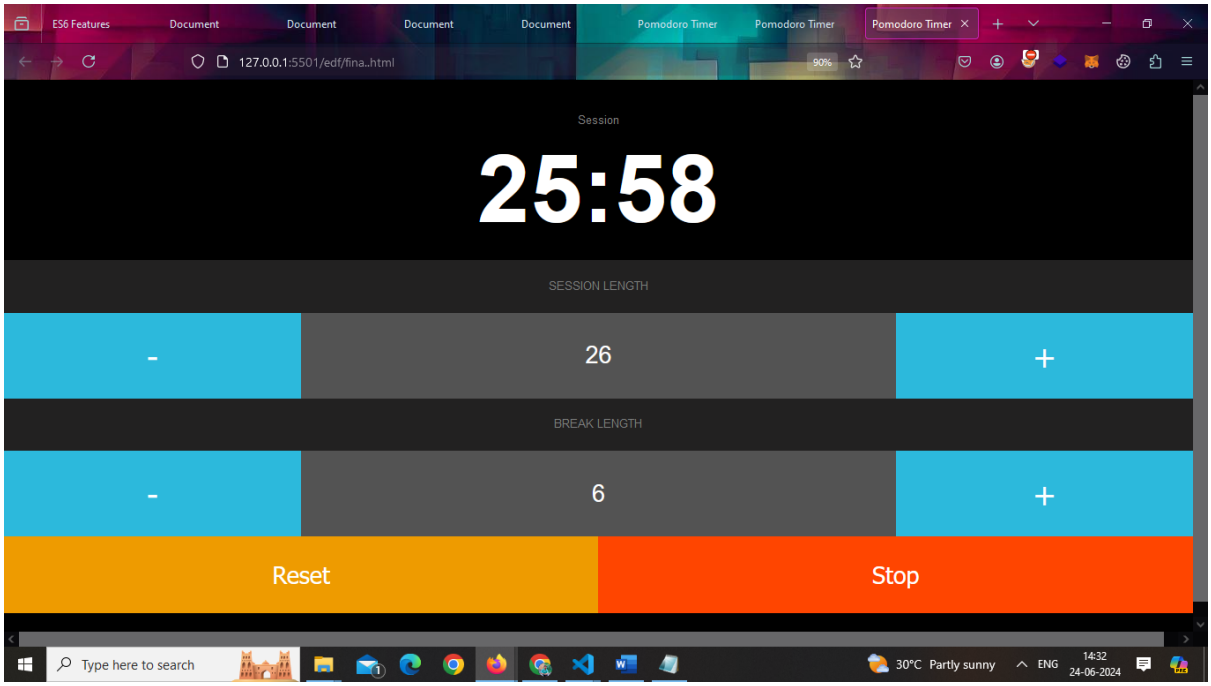
```
      .start {
        background-color: #32CD32;
      }

      .stop {
        background-color: #FF4500;
        display: none;
      }

      .reset-active {
        background-color: #ee9b00 !important;
        /* Yellow-Orange color */
      }
    </style>
</head>

<body>
    <header class="header">Session</header>
    <h1 class="timer">25:00</h1>
    <div class="labels">
      <div class="label">SESSION LENGTH</div>
    </div>

    <div class="controls">
      <button class="control2" id="decrementSession">-</button>
      <span class="control1" id="sessionLength">25</span>
      <button class="control2" id="incrementSession">+</button>
    </div>

    <div class="labels">
      <div class="label">BREAK LENGTH</div>
    </div>

    <div class="controls">
      <button class="control2" id="decrementBreak">-</button>
      <span class="control1" id="breakLength">5</span>
      <button class="control2" id="incrementBreak">+</button>
    </div>

    <div class="buttons">
      <button class="reset" disabled id="resetButton">Reset</button>
      <button class="start" id="startButton">Start</button>
      <button class="stop" id="stopButton">Stop</button>
    </div>


    <script>
      let sessionLength = 25;
      let breakLength = 5;
      let timerInterval;
```

```
    let isSession = true;
    let timeRemaining = sessionLength * 60;

    const timerDisplay = document.querySelector(".timer");
    const startButton = document.getElementById("startButton");
    const stopButton = document.getElementById("stopButton");
    const resetButton = document.getElementById("resetButton");
    const decrementSession = document.getElementById("decrementSession");
    const incrementSession = document.getElementById("incrementSession");
    const decrementBreak = document.getElementById("decrementBreak");
    const incrementBreak = document.getElementById("incrementBreak");
    const sessionLengthDisplay = document.getElementById("sessionLength");
    const breakLengthDisplay = document.getElementById("breakLength");

    function updateTimerDisplay() {
      const minutes = Math.floor(timeRemaining / 60);
      const seconds = timeRemaining % 60;
      timerDisplay.textContent = `${minutes.toString().padStart(2,
"0")}:${seconds.toString().padStart(2, "0")}`;
    }

    function startTimer() {
      timerInterval = setInterval(() => {
        timeRemaining--;
        updateTimerDisplay();
        if (timeRemaining === 0) {
          clearInterval(timerInterval);
          if (isSession) {
            isSession = false;
            timeRemaining = breakLength * 60;
            startTimer();
          } else {
            isSession = true;
            timeRemaining = sessionLength * 60;
            startTimer();
          }
        }
      }, 1000);

      startButton.style.display = "none";
      stopButton.style.display = "inline-block";
      resetButton.classList.add("reset-active"); // Add class to change reset button color
      resetButton.disabled = false;
    }

    function stopTimer() {
      clearInterval(timerInterval);
      startButton.style.display = "inline-block";
      stopButton.style.display = "none";
      resetButton.classList.remove("reset-active"); // Remove class if needed
      resetButton.disabled = false;
```

```
}

function resetTimer() {
  clearInterval(timerInterval);
  timeRemaining = sessionLength * 60;
  updateTimerDisplay();
  startButton.style.display = "inline-block";
  stopButton.style.display = "none";
  startButton.disabled = false;
  resetButton.classList.remove("reset-active"); // Remove class if needed
  resetButton.disabled = true;
}

function updateSessionLength(value) {
  sessionLength = value;
  sessionLengthDisplay.textContent = sessionLength;
  if (isSession) {
    timeRemaining = sessionLength * 60;
    updateTimerDisplay();
  }
}

function updateBreakLength(value) {
  breakLength = value;
  breakLengthDisplay.textContent = breakLength;
  if (!isSession) {
    timeRemaining = breakLength * 60;
    updateTimerDisplay();
  }
}

decrementSession.addEventListener("click", () => {
  if (sessionLength > 1) {
    updateSessionLength(sessionLength - 1);
  }
});

incrementSession.addEventListener("click", () => {
  updateSessionLength(sessionLength + 1);
});

decrementBreak.addEventListener("click", () => {
  if (breakLength > 1) {
    updateBreakLength(breakLength - 1);
  }
});

incrementBreak.addEventListener("click", () => {
  updateBreakLength(breakLength + 1);
});
```

```
    startButton.addEventListener("click", startTimer);
    stopButton.addEventListener("click", stopTimer);
    resetButton.addEventListener("click", resetTimer);

    updateTimerDisplay();
  </script>
</body>

</html>
```





5. Write a program to implement swap 1 to 9 numbers using drag and drop?

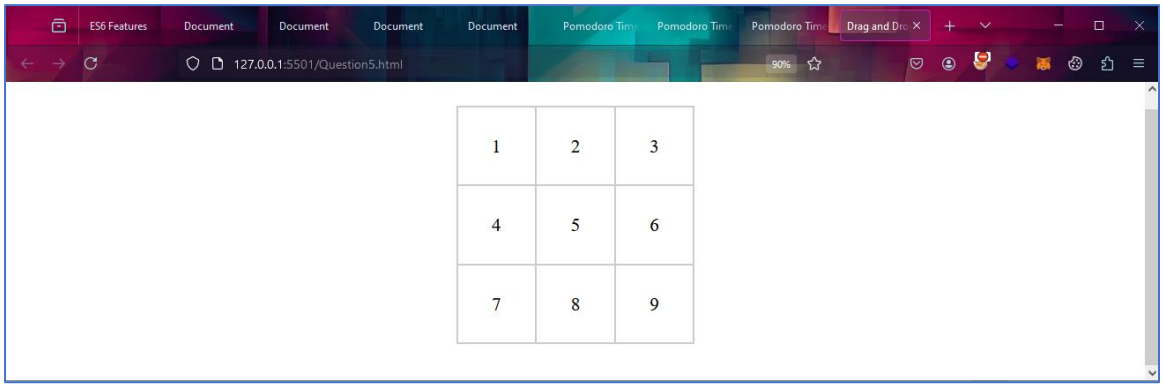
Code:

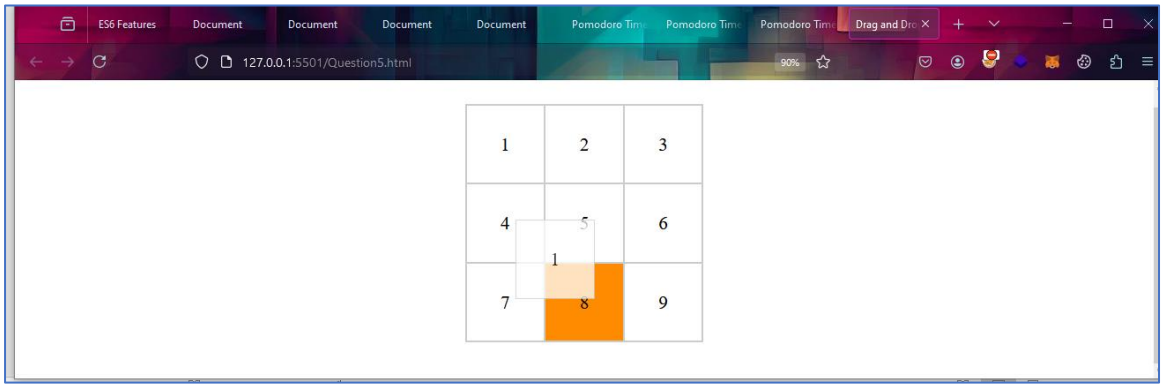
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Drag and Drop Number Swap</title>
   <link rel="stylesheet" href="styles.css">
   <style>
      .container {
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
      }

      .grid-container {
         display: grid;
         grid-template-columns: repeat(3, 100px);
         grid-template-rows: repeat(3, 100px);
         background-color: #f0f0f0;
         border: 1px solid #ccc;
         padding: 0px;
      }

      .grid-item {
         display: flex;
         justify-content: center;
         align-items: center;
         font-size: 24px;
         background-color: #ffffff;
         border: 1px solid #ccc;
         cursor: pointer;
      }

      .grid-item:hover {
         background-color: transparent rgb(255, 245, 245);
      }

      .drag-over {
         background-color: darkorange !important;
      }
   </style>
```

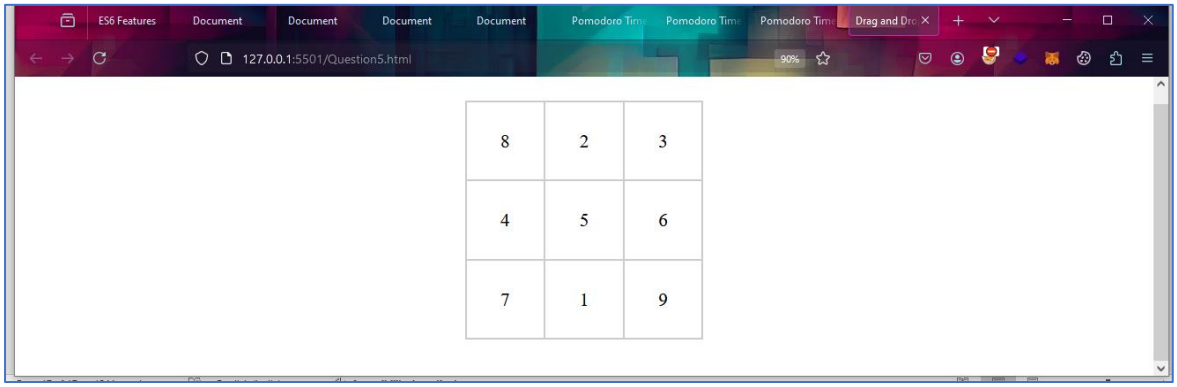
```
</head>

<body>
  <div class="container">
    <div class="grid-container" id="gridContainer">
      <div class="grid-item" id="cell1" draggable="true" ondragstart="drag(event)">1</div>
      <div class="grid-item" id="cell2" draggable="true" ondragstart="drag(event)">2</div>
      <div class="grid-item" id="cell3" draggable="true" ondragstart="drag(event)">3</div>
      <div class="grid-item" id="cell4" draggable="true" ondragstart="drag(event)">4</div>
      <div class="grid-item" id="cell5" draggable="true" ondragstart="drag(event)">5</div>
      <div class="grid-item" id="cell6" draggable="true" ondragstart="drag(event)">6</div>
      <div class="grid-item" id="cell7" draggable="true" ondragstart="drag(event)">7</div>
      <div class="grid-item" id="cell8" draggable="true" ondragstart="drag(event)">8</div>
      <div class="grid-item" id="cell9" draggable="true" ondragstart="drag(event)">9</div>
    </div>
  </div>

  <script>
    let draggedItem = null;

    function allowDrop(event) {
      event.preventDefault();
    }

    function drag(event) {
      draggedItem = event.target;
    }

    function dragEnter(event) {
      if (event.target.classList.contains('grid-item')) {
        event.target.classList.add('drag-over');
      }
    }

    function dragLeave(event) {
      if (event.target.classList.contains('grid-item')) {
        event.target.classList.remove('drag-over');
      }
    }

    function drop(event) {
      event.preventDefault();
      if (draggedItem !== null && event.target.classList.contains('grid-item')) {

        let temp = event.target.textContent;
        event.target.textContent = draggedItem.textContent;
        draggedItem.textContent = temp;
```

```
            event.target.classList.remove('drag-over');
            draggedItem = null;
        }
    }
    const gridItems = document.querySelectorAll('.grid-item');
    gridItems.forEach(item => {
        item.addEventListener('dragover', allowDrop);
        item.addEventListener('dragenter', dragEnter);
        item.addEventListener('dragleave', dragLeave);
        item.addEventListener('drop', drop);
    });
  </script>
</body>
</html>
```

Output:







6. Demonstrate all ES6 concepts with examples.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ES6 Features</title>
</head>

<body>
  <script>
    console.log();
    console.log("****************** let variable and constant const ************** ")
    let variable = 10; //let variable
    if (true) {
      let variable = 20; // let --> allows you to declare block-scoped variables
      console.log("let variable block-scoped (inside blocked)", variable); // print --> 20
    }
    console.log("Outside blocked let variable", variable); // print --> 10

    const constant = 30; // const variable --> it's allows you to declare block-scoped
constants
    console.log("constant", constant); // print --> 30
    // constant = 40;              // Error: Assignment to constant variable gives error for
reassigning the constant


    console.log();
    console.log("***************** Arrow Function ********************")
      //Arrow Function  --> (ES6 feature)
    const y = (a, b) => a * b;
    console.log("using Arrow Function", y(2, 3)); //6

    const greet = name => `Hello, ${name}`;
    console.log(greet('Alice')); // Hello, Alice

    console.log();
    console.log("***************** Rest & spread operator ********************")
      // rest operator collects multiple elements into an array
    function sum(...args) {
      return args.reduce((acc, curr) => acc + curr, 0);
    }
    console.log(sum(1, 2, 3)); // 6
```

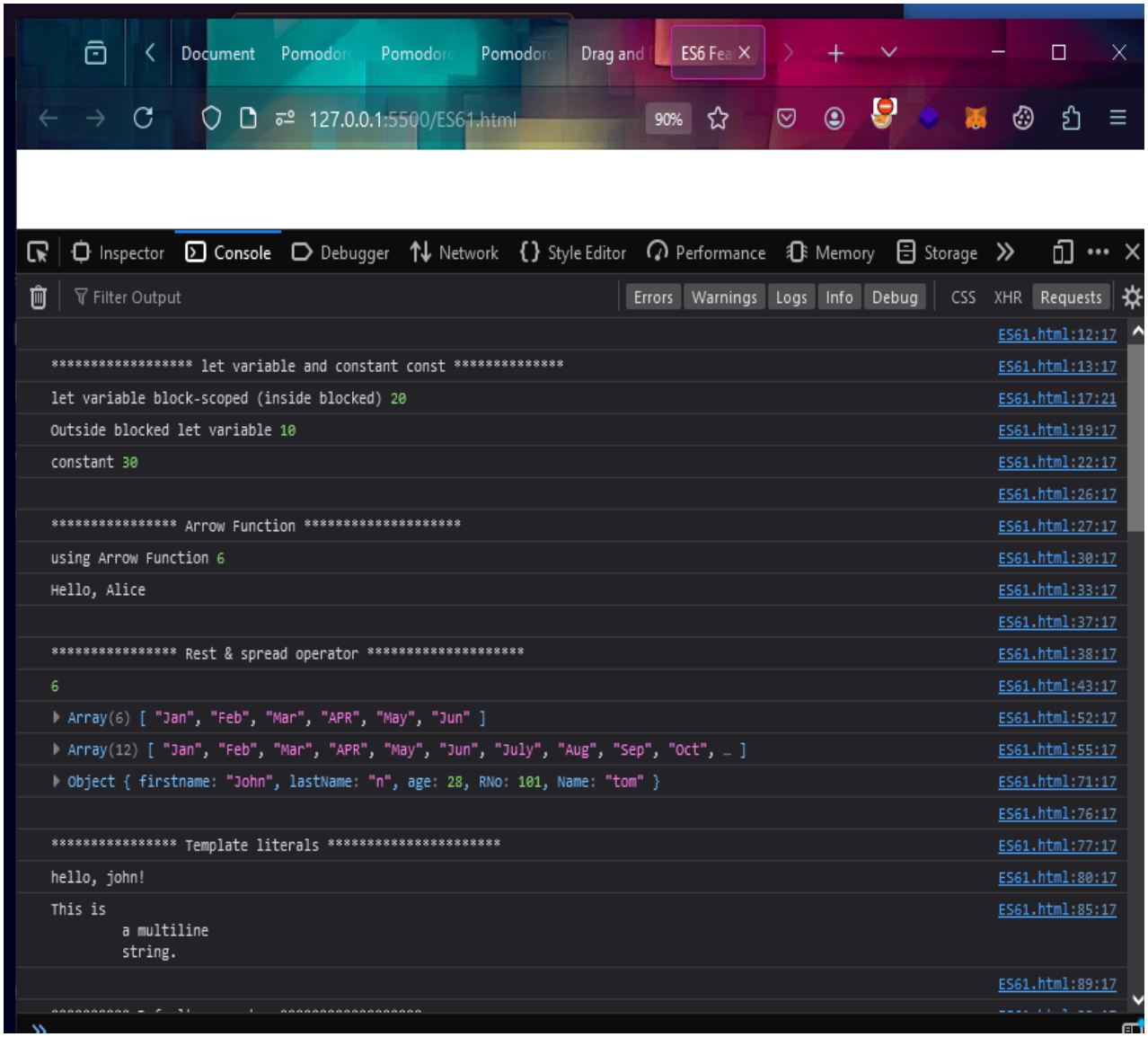
```
// Spread operator -the spread operatorspread elements of an array or object
const q1 = ["Jan", "Feb", "Mar"];
const q2 = ["APR", "May", "Jun"];
const q3 = ["July", "Aug", "Sep"];
const q4 = ["Oct", "Nov", "Dec"];

const Year1 = q1.concat(q2) //concat at a time 2 array only using concat
console.log(Year1);

const Year = [...q1, ...q2, ...q3, ...q4]; //concat aal four arrays
console.log(Year); // [ "Jan", "Feb", "Mar", "APR", "May", "Jun", "July", "Aug", "Sep",
"Oct", … ]

//objects
const Student1 = {
  firstname: "John",
  lastName: "n",
  age: 28,
};
const Student2 = {
  RNo: 101,
  Name: "tom",
};

const student = {...Student1,
  ...Student2
};
console.log(student); //Object { firstname: "John", lastName: "n", age: 28, RNo: 101,
Name: "tom" }


//Template literals allow for String imterpolation and multi-line strings
console.log();
console.log("**************** Template literals **********************");
const name = "john";
const greeting = `hello, ${name}!`;
console.log(greeting);

const multiline = `This is
 a multiline
 string. `
console.log(multiline);


console.log(); //default parameters allow you to set default values for function
parameters
console.log("********** Default parameter ******************");
```

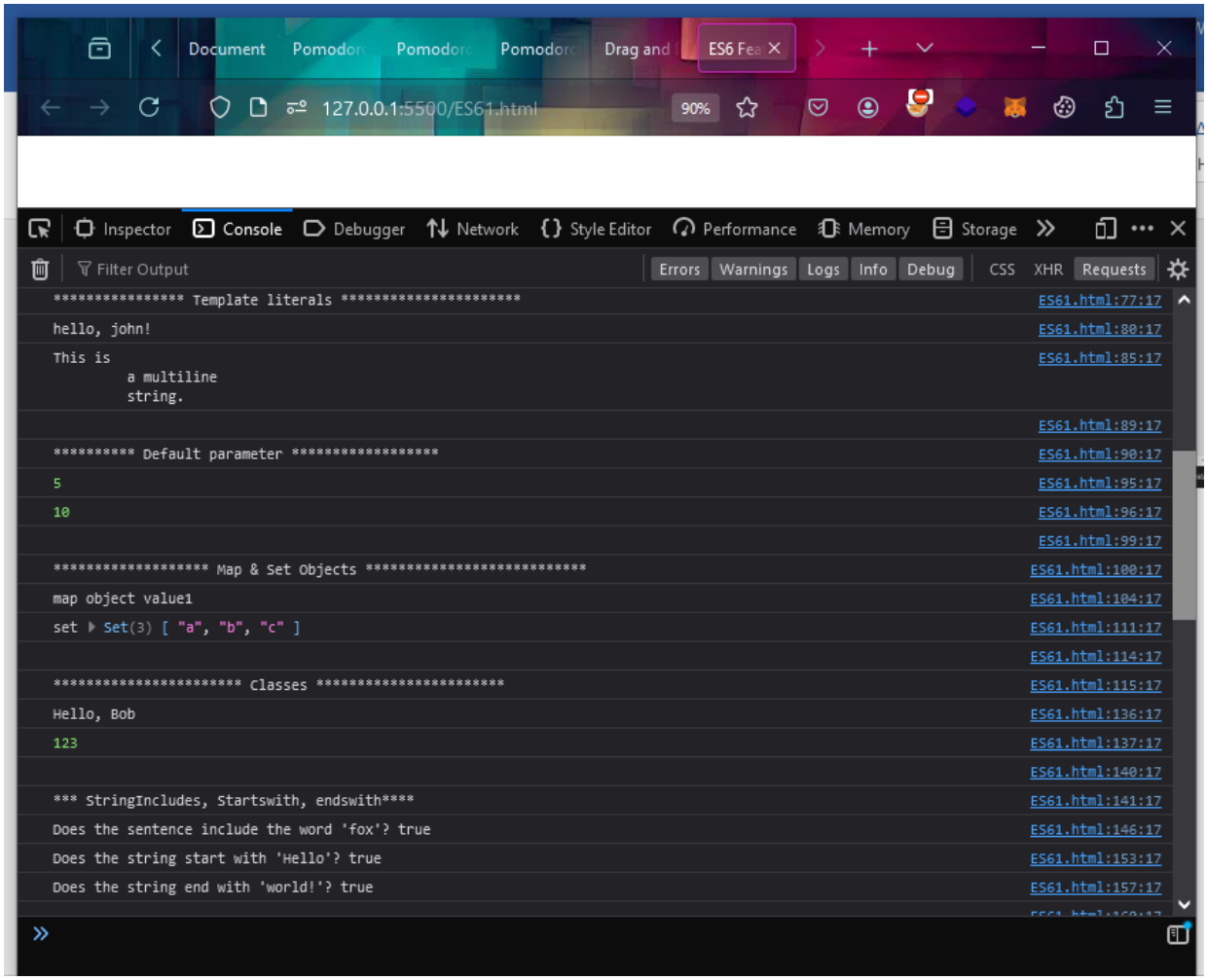
```
function multiply(a, b = 1) {
  return a * b;
}
console.log(multiply(5)); // 5
console.log(multiply(5, 2)); // 10


console.log();
console.log("******************** Map & Set Objects
**************************");
const map = new Map();
map.set('key1', 'value1');
map.set('key2', 'value2');
console.log("map object", map.get('key1'));

const letters = new Set(); // Create a Set
letters.add("a");
letters.add("b");
letters.add("c"); // Add some values to the Set

console.log("set", letters);


console.log(); // class are provide convienient way for creating object and handling
inheritance
console.log("************************ Classes ************************");
class Person {
  constructor(name) {
    this.name = name;
  }
  greet() {
    return `Hello, ${this.name}`;
  }
}

class Student extends Person {
  constructor(name, studentId) {
    super(name);
    this.studentId = studentId;
  }
  getStudentId() {
    return this.studentId;
  }
}

const student1 = new Student('Bob', 123);
```

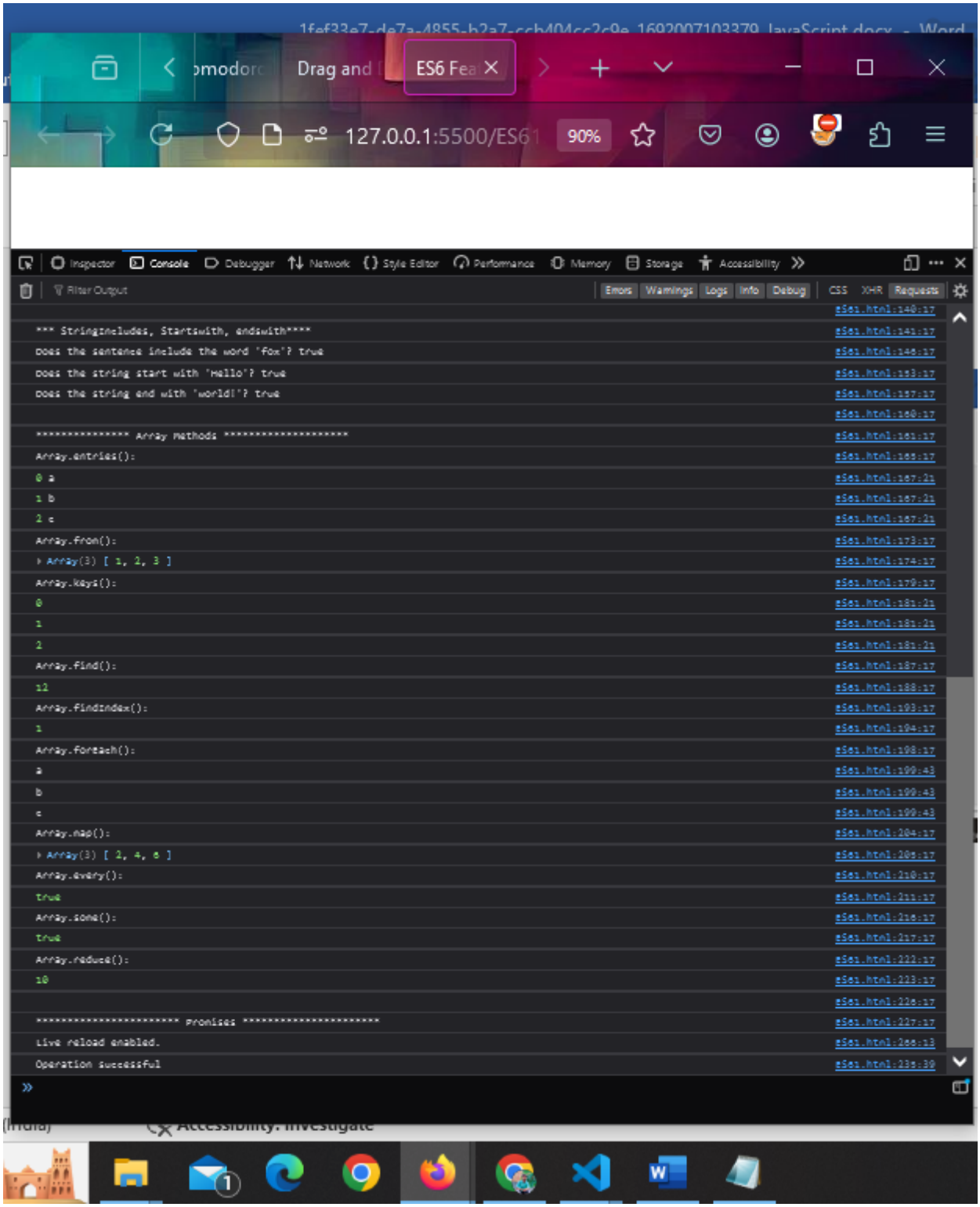
```
console.log(student1.greet()); // Hello, Bob
console.log(student1.getStudentId()); // 123


console.log();
console.log("*** StringIncludes, Startswith, endswith**** ");
// includes()
const sentence = "The quick brown fox jumps over the lazy dog.";
const word = "fox";

console.log(`Does the sentence include the word '${word}'?
${sentence.includes(word)}`);


// startsWith()
const str = "Hello, world!";
const prefix = "Hello";

console.log(`Does the string start with '${prefix}'? ${str.startsWith(prefix)}`);

// endsWith()
const suffix = "world!";
console.log(`Does the string end with '${suffix}'? ${str.endsWith(suffix)}`);


console.log();
console.log("*************** Array Methods ********************");
// Array.entries()
const array1 = ['a', 'b', 'c'];
const entries = array1.entries();
console.log('Array.entries():');
for (const [index, element] of entries) {
  console.log(index, element); // 0 'a', 1 'b', 2 'c'
}

// Array.from()
const set = new Set([1, 2, 3]);
const array2 = Array.from(set);
console.log('Array.from():');
console.log(array2); // [1, 2, 3]

// Array.keys()
const array3 = ['a', 'b', 'c'];
const keys = array3.keys();
console.log('Array.keys():');
for (const key of keys) {
  console.log(key); // 0, 1, 2
```

```
}

// Array.find()
const array4 = [5, 12, 8, 130, 44];
const found = array4.find(element => element > 10);
console.log('Array.find():');
console.log(found); // 12

// Array.findIndex()
const array5 = [5, 12, 8, 130, 44];
const foundIndex = array5.findIndex(element => element > 10);
console.log('Array.findIndex():');
console.log(foundIndex); // 1

// Array.forEach()
const array6 = ['a', 'b', 'c'];
console.log('Array.forEach():');
array6.forEach(element => console.log(element)); // 'a', 'b', 'c'

// Array.map()
const array7 = [1, 2, 3];
const mappedArray = array7.map(x => x * 2);
console.log('Array.map():');
console.log(mappedArray); // [2, 4, 6]

// Array.every()
const array8 = [1, 30, 39, 29, 10, 13];
const isBelow40 = array8.every(element => element < 40);
console.log('Array.every():');
console.log(isBelow40); // true

// Array.some()
const array9 = [1, 2, 3, 4, 5];
const hasEvenNumber = array9.some(element => element % 2 === 0);
console.log('Array.some():');
console.log(hasEvenNumber); // true

// Array.reduce()
const array10 = [1, 2, 3, 4];
const sum1 = array10.reduce((accumulator, currentValue) => accumulator +
currentValue, 0);
console.log('Array.reduce():');
console.log(sum1); // 10


console.log();
console.log("********************** Promises **********************");
```

```
    const asyncOperation = new Promise((resolve, reject) => {
      setTimeout(() => {
        resolve('Operation successful');
      }, 1000);
    });

    asyncOperation
      .then(response => console.log(response)) // Operation successful
      .catch(error => console.error(error));
  </script>
</body>

</html>
```




Document
Pomodor
Pomodor
Pomodor
Drag and
ES6 Fea
127.0.0.1:5500/ES61.html
90%
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
********** Template literals **********
ES61.html:77:17
hello, john!
ES61.html:80:17
This is
a multiline
string.
ES61.html:85:17
ES61.html:89:17
********** Default parameter **********
ES61.html:90:17
5
ES61.html:95:17
10
ES61.html:96:17
ES61.html:99:17
********** Map & Set Objects **********
ES61.html:100:17
map object value1
ES61.html:104:17
set ▶ Set(3) ["a", "b", "c"]
ES61.html:111:17
ES61.html:114:17
********** Classes **********
ES61.html:115:17
Hello, Bob
ES61.html:136:17
123
ES61.html:137:17
ES61.html:140:17
*** StringIncludes, Startswith, endswith****
ES61.html:141:17
Does the sentence include the word 'fox'? true
ES61.html:146:17
Does the string start with 'Hello'? true
ES61.html:153:17
Does the string end with 'world!'? true
ES61.html:157:17


Drag and
ES6 Fea
127.0.0.1:5500/ES61
90%
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
*** StringIncludes, Startswith, endswith****
Does the sentence include the word 'fox'? true
Does the string start with 'Hello'? true
Does the string end with 'world!'? true
********** Array Methods **********
Array.entries():
0 a
1 b
2 c
Array.from():
▸ Array(3) [1, 2, 3]
Array.keys():
0
1
2
Array.find():
12
Array.findIndex():
1
Array.forEach():
a
b
c
Array.map():
▸ Array(3) [2, 4, 6]
Array.every():
true
Array.some():
true
Array.reduce():
10
********** promises **********
Live reload enabled.
Operation successful