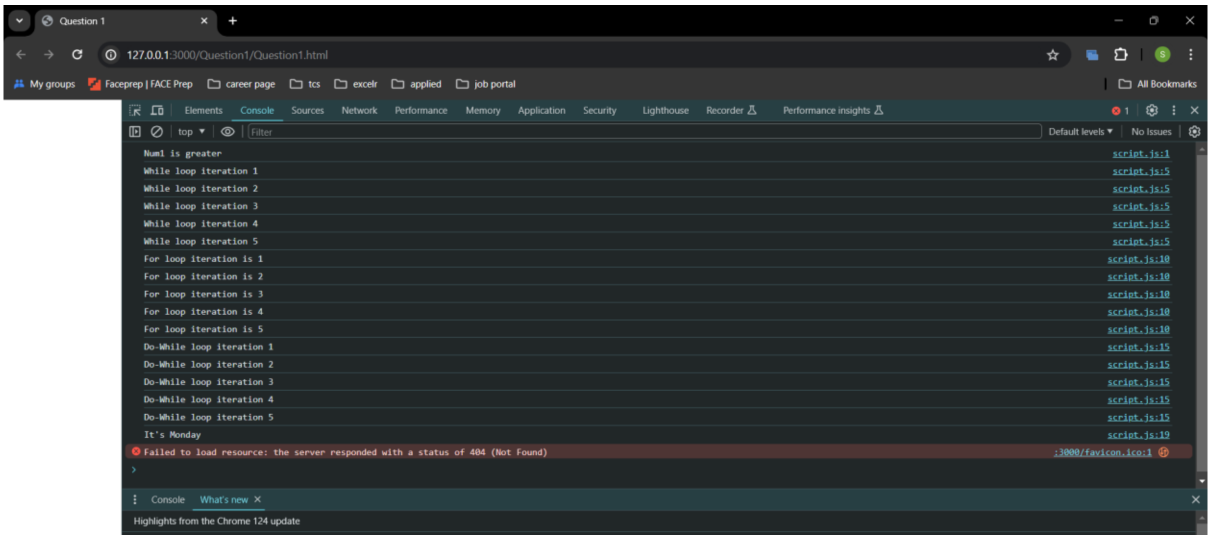
JavaScript Assignment

1.

```html
 <!DOCTYPE html>

<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 1</title>
</head>

<body>

  <script src="script.js"></script>

</body>

</html>
```

```javascript
console.log("Num1 is greater\n");

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

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

let k = 1;
do {
  console.log("Do-While loop iteration " + k);
  k++;
} while (k <= 5);

console.log("It's Monday");
```

Output:



```
Num1 is greater                                                                  script.js:1
While loop iteration 1                                                           script.js:5
While loop iteration 2                                                           script.js:5
While loop iteration 3                                                           script.js:5
While loop iteration 4                                                           script.js:5
While loop iteration 5                                                           script.js:5
For loop iteration is 1                                                          script.js:10
For loop iteration is 2                                                          script.js:10
For loop iteration is 3                                                          script.js:10
For loop iteration is 4                                                          script.js:10
For loop iteration is 5                                                          script.js:10
Do-While loop iteration 1                                                        script.js:15
Do-While loop iteration 2                                                        script.js:15
Do-While loop iteration 3                                                        script.js:15
Do-While loop iteration 4                                                        script.js:15
Do-While loop iteration 5                                                        script.js:15
It's Monday                                                                      script.js:19
Failed to load resource: the server responded with a status of 404 (Not Found)   :3000/favicon.ico:1
>
```

2.

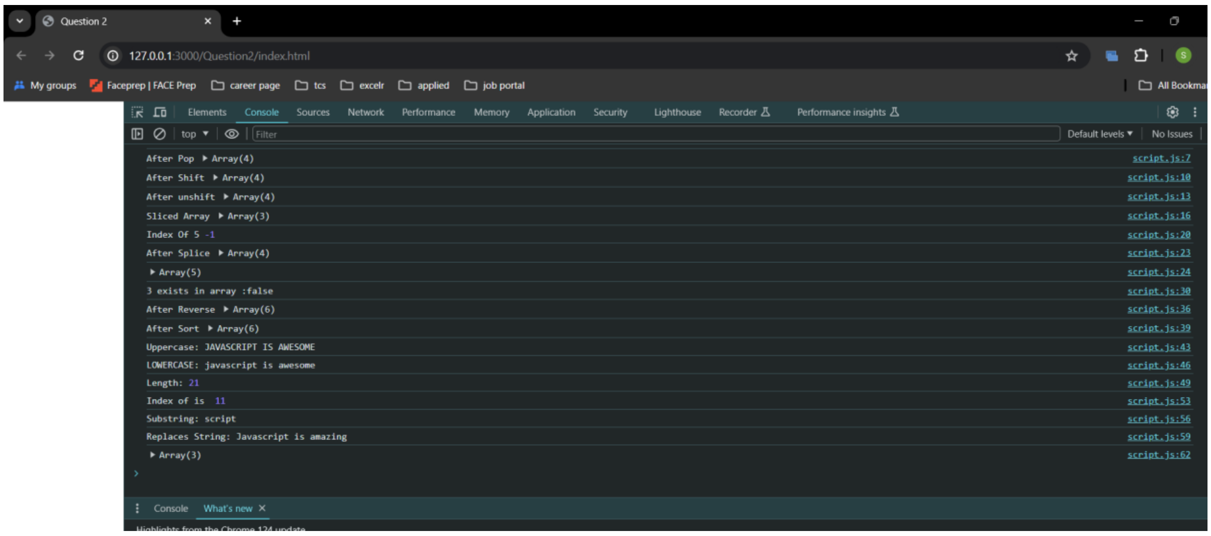
```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 2</title>
</head>

<body>
  <script src="./script.js"></script>
</body>

</html>
```

```javascript
const numbers = [1, 2, 3, 4, 5,6];

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

numbers.pop();
console.log("After Pop", numbers);

numbers.shift();
```

```
console.log("After Shift", numbers);

numbers.unshift(0);
console.log("After unshift", numbers);

const slicedArray = numbers.slice(1,4);
console.log("Sliced Array", slicedArray);

const startIndex = numbers.indexOf(1, 1);
const index = numbers.indexOf(5, startIndex);
console.log("Index Of 5", index);

const removedElements = numbers.splice(0, 5, 0, 8, 9);
console.log("After Splice", numbers);
console.log(removedElements);

const numberToCheck = 3;
if (numbers.includes(numberToCheck)) {
  console.log(`3 exists in array :true`);
} else {
  console.log(`3 exists in array :false`);
}

const originalArray = [0, 8, 9, 4, 5, 6];

const reversed = originalArray.reverse();
console.log("After Reverse", reversed);

const sortedArray = originalArray.sort((a, b) => a - b);
console.log("After Sort", sortedArray);

const originalString = "Javascript is awesome";
const uppercaseString = originalString.toUpperCase();
console.log("Uppercase:", uppercaseString);

const lowercaseString = originalString.toLowerCase();
console.log("LOWERCASE:", lowercaseString);

const length = originalString.length;
console.log("Length:", length);

const substring = "is";
const index1 = originalString.indexOf(substring);
console.log("Index of is ", index1);

const substring1 = originalString.substring(4, 10);
console.log("Substring:", substring1);
```

```
const modifiedString = originalString.replace("awesome", "amazing");
console.log("Replaces String:",modifiedString);

const wordArray = originalString.split(" ");
console.log(wordArray);
```

Output:



```
After Pop ▸ Array(4)                                  script.js:7
After Shift ▸ Array(4)                                script.js:10
After unshift ▸ Array(4)                              script.js:13
Sliced Array ▸ Array(3)                               script.js:16
Index Of 5 -1                                         script.js:20
After Splice ▸ Array(4)                               script.js:23
 ▸ Array(5)                                           script.js:24
3 exists in array :false                              script.js:30
After Reverse ▸ Array(6)                              script.js:36
After Sort ▸ Array(6)                                 script.js:39
Uppercase: JAVASCRIPT IS AWESOME                      script.js:43
LOWERCASE: javascript is awesome                      script.js:46
Length: 21                                            script.js:49
Index of is  11                                       script.js:53
Substring: script                                     script.js:56
Replaces String: Javascript is amazing                script.js:59
 ▸ Array(3)                                           script.js:62
```

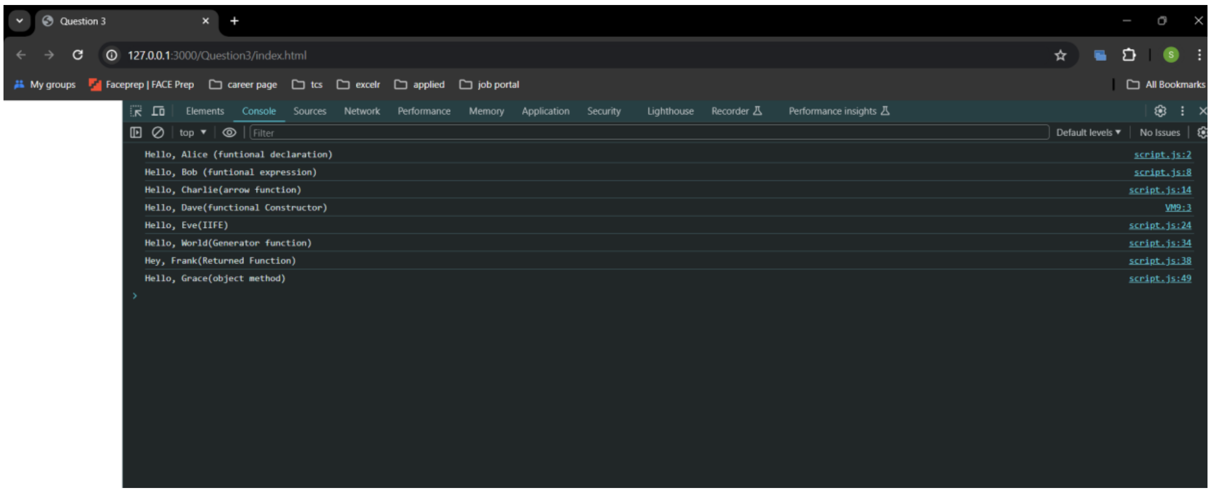
3.

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

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 3</title>
</head>

<body>
  <script src="script.js"></script>
</body>

</html>
```

```
function greet(name) {
  console.log(`Hello, ${name}`);
}

greet("Alice (funtional declaration)");
```

```
const greet1= function(name) {
  console.log(`Hello, ${name}`);
};

greet1("Bob (funtional expression)");

const greet3= (name) => {
  console.log(`Hello, ${name}`);
};

greet3("Charlie(arrow function)");

const greet4 = new Function("name", "console.log(`Hello, ${name}`);");

greet4("Dave(functional Constructor)");

(function(name) {
  console.log(`Hello, ${name}`);
})("Eve(IIFE)");

function* helloGenerator() {
  yield "Hello,";
  yield " World(Generator function)";
}

const generator = helloGenerator();

console.log(generator.next().value + generator.next().value);

function createGreeting(name) {
  return function() {
    console.log(`Hey, ${name}`);
  };
}

const greetFrank = createGreeting("Frank(Returned Function)");

greetFrank();

const person = {
  name: "Grace",
  sayHello: function() {
    console.log(`Hello, ${this.name}(object method)`);
  }
};

person.sayHello();
```

Output:



```
Hello, Alice (funtional declaration)                script.js:2
Hello, Bob (funtional expression)                   script.js:8
Hello, Charlie(arrow function)                      script.js:14
Hello, Dave(functional Constructor)                 VM9:3
Hello, Eve(IIFE)                                    script.js:24
Hello, World(Generator function)                    script.js:34
Hey, Frank(Returned Function)                       script.js:38
Hello, Grace(object method)                         script.js:49
```

4.

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

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 4</title>
  <link rel="stylesheet" href="./style.css" />
  <script src="./script.js"></script>
</head>

<body>
  <div class="container1">
    <h6 id="heading-container">SESSION</h6>
    <h1 id="timer">14:55</h1>
  </div>

  <div class="container2" style="background-color: #222222;">
    <h1 style="padding: 20px; font-size: 15px; font-family: 'Fira Sans', sans-
serif; color: grey;">SESSION LENGTH</h1>
  </div>

  <div class="main-container">
    <div class="subcontainer1" style="background-color: #3486f1;"
onclick="changeSessionLength(-1)">
      <h1>-</h1>
    </div>
    <div class="subcontainer2" style="background-color: grey;"
id="sessionLength">16</div>
    <div class="subcontainer3" style="background-color: #3486f1;"
onclick="changeSessionLength(1)">
      <h1>+</h1>
    </div>
  </div>

  <div class="container2" style="background-color: #222222;">
    <h1 style="padding: 20px; font-size: 15px; font-family: 'Fira Sans', sans-
serif; color: grey;">BREAK LENGTH</h1>
  </div>

  <div class="main-container">
    <div class="subcontainer1" style="background-color: #3486f1;"
onclick="changeBreakLength(-1)">
      <h1>-</h1>
    </div>
    <div class="subcontainer2" style="background-color: grey;"
id="breakLength">5</div>
    <div class="subcontainer3" style="background-color: #3486f1;"
onclick="changeBreakLength(1)">
      <h1>+</h1>
    </div>
  </div>

  <div class="main-container1">
    <div class="btn" onclick="resetTimer()">
      <h1>Reset</h1>
    </div>
    <div class="btn" id="startStop" onclick="startStopTimer()">
      <h1>Stop</h1>
    </div>
  </div>
</body>

</html>
```

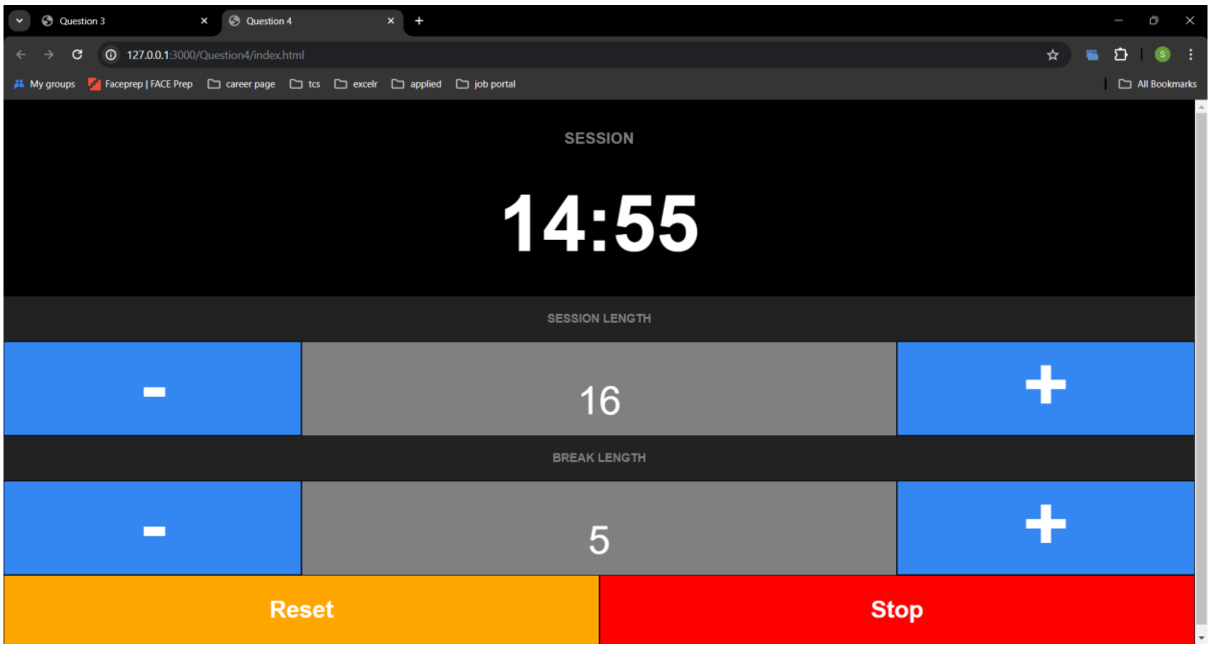
```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: "Fira Sans", sans-serif;
```

```css
}

.container1 {
  background-color: black;
  height: 250px;
  margin-top: 0;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  border: 1px solid black;
}

h1 {
  color: white;
  text-align: center;
  font-size: 100px;
  padding-bottom: 10px;
  font-family: sans-serif;
}

#timer {
  font-size: 100px;
  padding-top: 40px;
}

.main-container {
  display: flex;
  height: 120px;
}

#heading-container {
  text-align: center;
  color: grey;
  font-size: 20px;
  margin-top: 12px;
}

.container2 {
  background-color: #222222;
  text-align: center;
}

.container2 h1 {
  padding: 20px;
  font-size: 15px;
  font-family: "Fira Sans", sans-serif;
  color: grey;
```

```css
}

.subcontainer1,
.subcontainer3 {
  border: 1px solid black;
  flex: 1;
  display: flex;
  justify-content: center;
  align-items: center;
}

.subcontainer2 {
  border: 1px solid black;
  flex: 2;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 50px;
  color: white;
  padding-top: 30px;
  height: 121px;
}

.main-container1 {
  display: flex;
  justify-content: space-around; /* Align buttons with space around them */
  height: 95px;
}

.btn {
  border: 1px solid black;
  flex: 1;
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100px;
  background-color: orange;
}

#startStop {
  background-color: red;
  color: white;
}

.btn h1 {
  font-size: 30px;
  color: white;
}
```

```javascript
let sessionLength = 16;
let breakLength = 5;
let timer;
let isRunning = false;
let minutes;
let seconds;
let mode = "session"; // Added mode variable

function updateTimerDisplay() {
  document.getElementById("timer").textContent = formatTime(minutes, seconds);
}

function formatTime(min, sec) {
  return `${String(min).padStart(2, "0")}:${String(sec).padStart(2, "0")}`;
}

function changeSessionLength(amount) {
  sessionLength += amount;
  if (sessionLength < 1) {
    sessionLength = 1;
  }
  document.getElementById("sessionLength").textContent = sessionLength;
}

function changeBreakLength(amount) {
  breakLength += amount;
  if (breakLength < 1) {
    breakLength = 1;
  }
  document.getElementById("breakLength").textContent = breakLength;
}

function startTimer() {
  if (!isRunning) {
    minutes = sessionLength;
    seconds = 0;
    updateTimerDisplay();
    timer = setInterval(updateTimer, 1000);
    isRunning = true;
    mode = "session";
    document.getElementById("startStop").textContent = "Stop";
    document.getElementById("startStop").style.backgroundColor = "red";
  }
}

function stopTimer() {
  clearInterval(timer);
  isRunning = false;
  document.getElementById("startStop").textContent = "Start";
  document.getElementById("startStop").style.backgroundColor = "green";
}

function startStopTimer() {
  if (isRunning) {
    stopTimer();
  } else {
    startTimer();
  }
}

function updateTimer() {
  if (minutes === 0 && seconds === 0) {
    if (mode === "session") {
      minutes = breakLength;
      mode = "break";
    } else {
      minutes = sessionLength;
      mode = "session";
    }
    updateTimerDisplay();
    alert("Timer completed!");
  } else {
    if (seconds === 0) {
      minutes--;
      seconds = 59;
    } else {
      seconds--;
    }
    updateTimerDisplay();
  }
}

function resetTimer() {
  stopTimer();
  minutes = sessionLength;
  seconds = 0;
  updateTimerDisplay();
  document.getElementById("startStop").textContent = "Start";
  document.getElementById("startStop").style.backgroundColor = "green";
}
```

Output:



5.

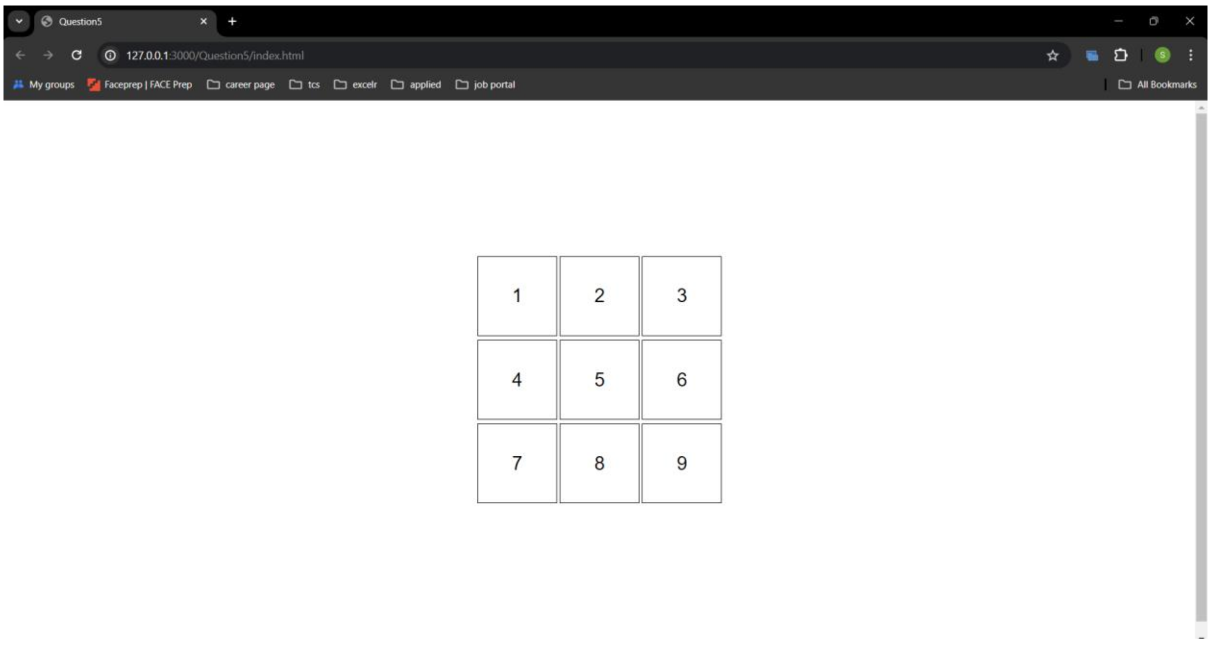
```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Question5</title>
</head>

<body>
    <div class="container">
        <div class="grid">
            <div class="cell" draggable="true" data-number="1">1</div>
            <div class="cell" draggable="true" data-number="2">2</div>
            <div class="cell" draggable="true" data-number="3">3</div>
            <div class="cell" draggable="true" data-number="4">4</div>
            <div class="cell" draggable="true" data-number="5">5</div>
            <div class="cell" draggable="true" data-number="6">6</div>
            <div class="cell" draggable="true" data-number="7">7</div>
            <div class="cell" draggable="true" data-number="8">8</div>
            <div class="cell" draggable="true" data-number="9">9</div>
        </div>
    </div>
    <script src="script.js"></script>
</body>
```

```
</html>
```

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background-color: #fff;
}

.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.grid {
  display: grid;
  grid-template-columns: repeat(3, 100px);
  grid-gap: 5px;
}

.cell {
  width: 100px;
  height: 100px;
  background-color: #fff;
  border: 1px solid #000;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 24px;
  cursor: pointer;
  transition: background-color 0.3s ease-in-out, color 0.3s ease-in-out;
}

.cell.dragged {
  background-color: rgb(208, 120, 42);
  /* Orange */
}
```

```
const cells = document.querySelectorAll(".cell");
let draggedCell = null;

cells.forEach((cell) => {
    cell.addEventListener("dragstart", handleDragStart);
    cell.addEventListener("dragover", handleDragOver);
    cell.addEventListener("dragenter", handleDragEnter);
    cell.addEventListener("dragleave", handleDragLeave);
    cell.addEventListener("drop", handleDrop);
    cell.addEventListener("dragend", handleDragEnd);
});

function handleDragStart(e) {
    draggedCell = this;
    e.dataTransfer.effectAllowed = "move";
    e.dataTransfer.setData("text/html", this.innerHTML);
}

function handleDragOver(e) {
    if (e.preventDefault) {
        e.preventDefault();
    }
    e.dataTransfer.dropEffect = "move";
    return false;
}

function handleDragEnter(e) {
    if (this !== draggedCell) {
        this.classList.add("over");
    }
}

function handleDragLeave(e) {
    if (this !== draggedCell) {
        this.classList.remove("over");
    }
}

function handleDrop(e) {
    if (e.stopPropagation) {
        e.stopPropagation();
    }

    if (draggedCell !== this) {
        // Swap the innerHTML and data-number attributes
        const tempInnerHTML = this.innerHTML;
        const tempDataNumber = this.getAttribute("data-number");

        this.innerHTML = draggedCell.innerHTML;
        this.setAttribute("data-number", draggedCell.getAttribute("data-
number"));

        draggedCell.innerHTML = tempInnerHTML;
        draggedCell.setAttribute("data-number", tempDataNumber);
        // Change the color of the dragged cell
        draggedCell.classList.remove("dragged");
        this.classList.add("dragged");
    }

    return false;
}

function handleDragEnd(e) {
    cells.forEach((cell) => {
        cell.classList.remove("over");
    });
}
```

Output:



6.

let and const

let and const provide block-scoped variable declarations, unlike var which is function-scoped.

javascript

let x = 10;

if (true) {

let x = 20;

```
console.log(x);
}
console.log(x);

const PI = 3.14;
PI = 3;
```

Arrow Functions

Arrow functions provide a concise syntax for defining functions.

```
function add(a, b) {
return a + b;
}

const add = (a, b) => a + b;
console.log(add(2, 3));
```

Template Literals

Template literals allow embedding expressions inside strings using backticks (`).

```
const name = 'Alice';
const greeting = Hello, ${name}!;
console.log(greeting);
```

Destructuring

Destructuring allows extracting values from arrays or objects into distinct variables.

```
const numbers = [1, 2, 3];
const [a, b, c] = numbers;
console.log(a, b, c);

const person = { firstName: 'John', lastName: 'Doe' };
const { firstName, lastName } = person;
console.log(firstName, lastName);
```

Spread and Rest Operators

The spread operator (...) spreads elements of an iterable (like an array) into individual elements. The rest parameter (...) gathers individual elements into an array.

```
const arr1 = [1, 2, 3];

const arr2 = [...arr1, 4, 5];

console.log(arr2);


const sum = (...args) => args.reduce((acc, val) => acc + val, 0);

console.log(sum(1, 2, 3));
```

Classes

ES6 introduced class syntax for defining JavaScript classes.

```
class Rectangle {

constructor(width, height) {

this.width = width;

this.height = height;

}

area() {

return this.width * this.height;

}

}


const rect = new Rectangle(5, 10);

console.log(rect.area());
```

Modules

ES6 modules allow organizing code into reusable components.

```
export const add = (a, b) => a + b;

export const multiply = (a, b) => a * b;
```

```
import { add, multiply } from './math.js';
console.log(add(2, 3));
console.log(multiply(2, 3));
```

Promises

Promises provide a cleaner way to work with asynchronous code.

```
const fetchData = () => {
return new Promise((resolve, reject) => {
setTimeout(() => {
resolve('Data fetched successfully!');
}, 2000);
});
};

fetchData()
.then((data) => console.log(data))
.catch((error) => console.error(error));
```

Default Parameters

ES6 allows defining default parameter values for functions.

javascript

Copy code

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

greet();
greet('Alice');
```

Async/Await

Async/await simplifies working with promises, making asynchronous code look synchronous.

```
const fetchData = () => {
return new Promise((resolve, reject) => {
setTimeout(() => {
resolve('Data fetched successfully!');
}, 2000);
});
};

const getData = async () => {
try {
const data = await fetchData();
console.log(data);
} catch (error) {
console.error(error);
}
};

getData();
```

these are the examples of ES6 in javascript