Lesson 04 Demo 03

Implementing String Manipulation and Optimization

Objective: To implement string manipulation and optimization techniques in JavaScript for efficient and secure handling of data

Tools required: Visual Studio Code and Node.js

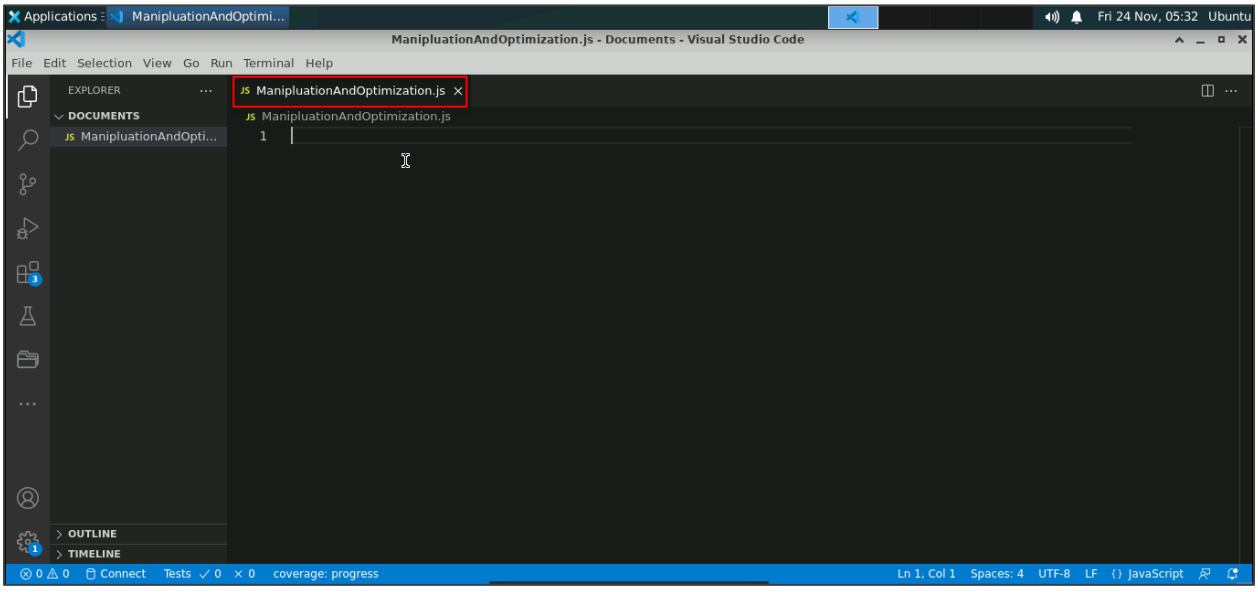
Prerequisites: A basic understanding of string operations in JavaScript

Steps to be followed:

1. Create and execute the JS file

Step 1: Create and execute the JS file

1.1 Open the Visual Studio Code editor and create a JavaScript file named **ManipulationAndOptimization.js**



1.2 Add the following code to the **ManipulationAndOptimization.js** file:

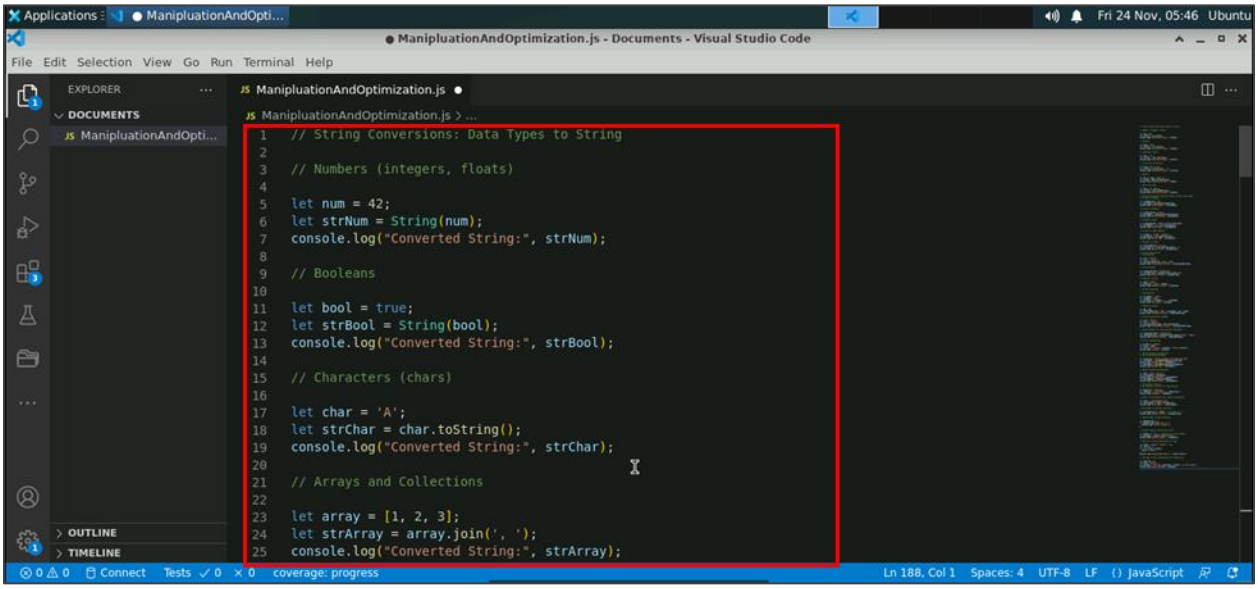
```
// String Conversions: Data Types to String
// Numbers (integers, floats)

let num = 42;
let strNumm = String(num);
console.log("Converted String:", strNumm);

// Booleans

let bool = true;
let strBool1 = String(bool);
console.log("Converted String:", strBool1);

// Characters (chars)

let char = 'A';
let strChar = char.toString();
console.log("Converted String:", strChar);

// Arrays and Collections

let array = [1, 2, 3];
let strArrays = array.join(', ');
console.log("Converted String:", strArrays);

// Objects

let obj = { key: 'value' };
let strObj = JSON.stringify(obj);
console.log("Converted String:", strObj);

// Dates and Times

let date = new Date();
let strDate1 = date.toISOString();
console.log("Converted String:", strDate1);

// String Conversions: Parsing Strings to Other Data Types
// Strings to Numbers

let strNum = "42";
```

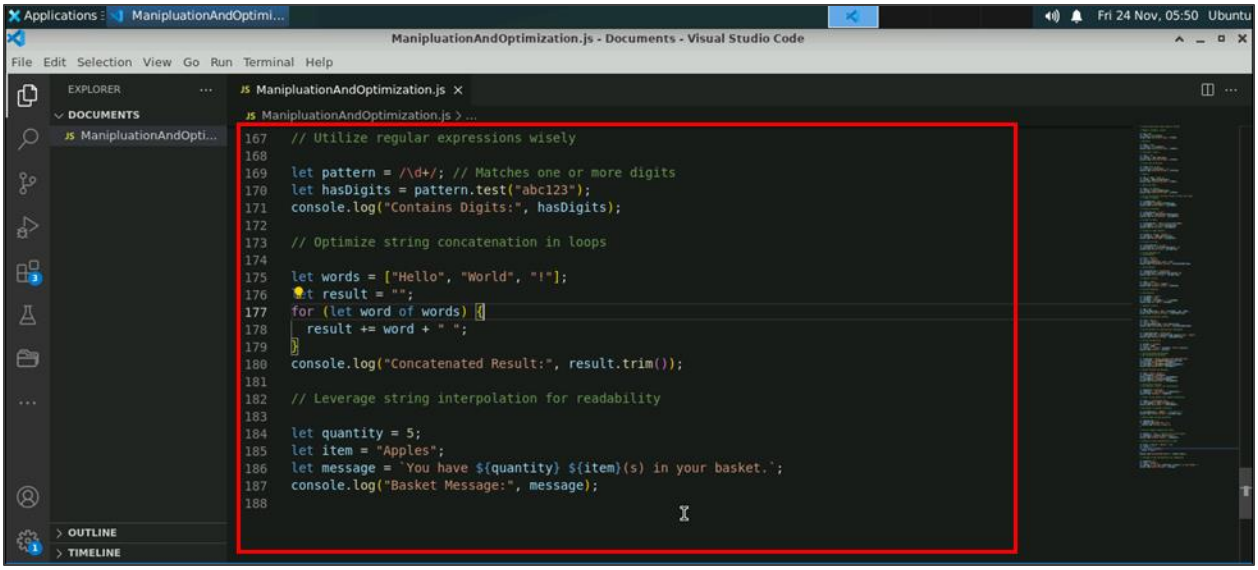
```
let parsedNum = parseInt(strNum);

console.log("Parsed Number:", parsedNum);

// Strings to Boolean

let strBool = "true";
let parsedBool = JSON.parse(strBool);
console.log("Parsed Boolean:", parsedBool);

// Strings to Dates

let strDate = "2023-11-24T12:00:00.000Z";
let parsedDate = new Date(strDate);
console.log("Parsed Date:", parsedDate);

// Strings to JSON (Objects)

let strJson = '{"key": "value"}';
let parsedJson = JSON.parse(strJson);
console.log("Parsed JSON:", parsedJson);

// Strings to Arrays

let strArray = "1,2,3";
let parsedArray = strArray.split(',');
console.log("Parsed Array:", parsedArray);

// String Immutability
// Concatenation

let strOne = "Hello";
let strTwo = " World!";
let concatenatedStringg = str1 + str2;
console.log("Concatenated String:", concatenatedStringg);

// String Methods

let originalStringg = "JavaScript";
let slicedString = originalStringg.slice(0, 4);
console.log("Sliced String:", slicedString);

// Template literals
```

```
let name = "John";

let greeting = `Hello, ${name}!`;
console.log("Template Literal:", greeting);

// String Formatting
// Concatenation

let firstName = "John";
let lastName = "Doe";
let fullName = firstName + " " + lastName;
console.log("Full Name:", fullName);

// Template literals

let age = 30;
let formattedString = `Name: ${fullName}, Age: ${age}`;
console.log("Formatted String:", formattedString);

// String concatenation methods

let str1 = "Hello";
let str2 = " World!";
let concatenatedString = str1.concat(str2);
console.log("Concatenated String:", concatenatedString);

// String methods for substring and replacement

let originalString = "JavaScript";
let replacedString = originalString.replace("Java", "Type");
console.log("Replaced String:", replacedString);

// String interpolation

let product = "Laptop";
let price = 1000;
let invoice = `Product: ${product}, Price: $${price}`;
console.log("Invoice:", invoice);

// String Encoding and Decoding
// URL encoding and decoding

let urlString = "https://example.com/?name=John Doe";
let encodedUrl = encodeURIComponent(urlString);
```

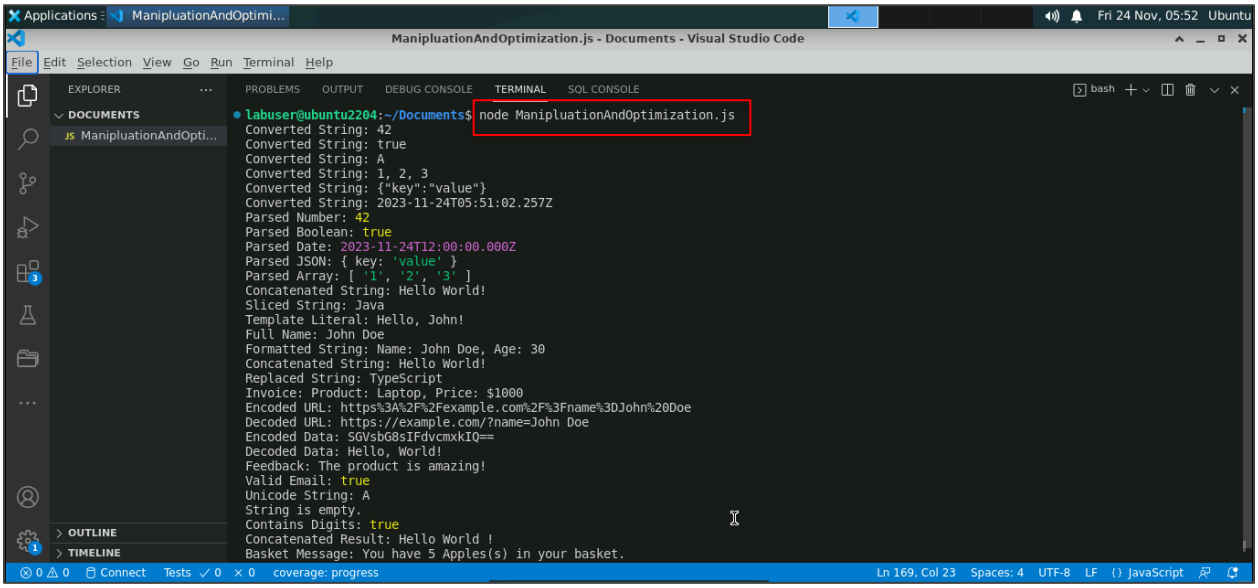
```
console.log("Encoded URL:", encodedUrl);

let decodedUrl = decodeURIComponent(encodedUrl);
console.log("Decoded URL:", decodedUrl);

// Base64 encoding and decoding

let data = "Hello, World!";
let encodedData = btoa(data);
console.log("Encoded Data:", encodedData);
let decodedData = atob(encodedData);
console.log("Decoded Data:", decodedData);

// String Best Practices
// Use template literals for concatenation

let adjective = "amazing";
let feedback = `The product is ${adjective}!`;
console.log("Feedback:", feedback);

// Prefer string methods over regular expressions

let email = "user@example.com";
let isValidEmail = email.includes("@");
console.log("Valid Email:", isValidEmail);

// Be mindful of Unicode characters

let unicodeString = "\u0041"; // Unicode for 'A'
console.log("Unicode String:", unicodeString);

// Handle empty strings gracefully

let emptyString = "";
if (emptyString.trim() === "") {
  console.log("String is empty.");
}

// Sanitize user input for security

let userInput = "<script>alert('Hello');</script>";
let sanitizedInput = DOMPurify.sanitize(userInput);
console.log("Sanitized Input:", sanitizedInput);
```

```
// Utilize regular expressions wisely

let pattern = /\d+/; // Matches one or more digits
let hasDigits = pattern.test("abc123");
console.log("Contains Digits:", hasDigits);

// Optimize string concatenation in loops

let words = ["Hello", "World", "!"];
let result = "";
for (let word of words) {
  result += word + " ";
}
console.log("Concatenated Result:", result.trim());

// Leverage string interpolation for readability

let quantity = 5;
let item = "Apples";
let message = `You have ${quantity} ${item}(s) in your basket.`;
console.log("Basket Message:", message);
```





1.3 Save the file and execute the below command in the terminal:
node ManipulationAndOptimization.js



The given code demonstrates converting data types to strings and parsing strings back to various data types. It explores string manipulation techniques, including concatenation, slicing, and encoding or decoding. Additionally, it emphasizes best practices such as using template literals, string methods, and considerations for Unicode characters and empty strings.

By following these steps, you have successfully demonstrated effective string manipulation and optimization techniques in JavaScript, ensuring efficient and secure handling of data.