Lesson 04 Demo 09

Demonstrating Advance String Operations

Objective: To demonstrate advanced JavaScript string operations for enhanced manipulation and iteration

Tools required: Visual Studio Code and Node.js

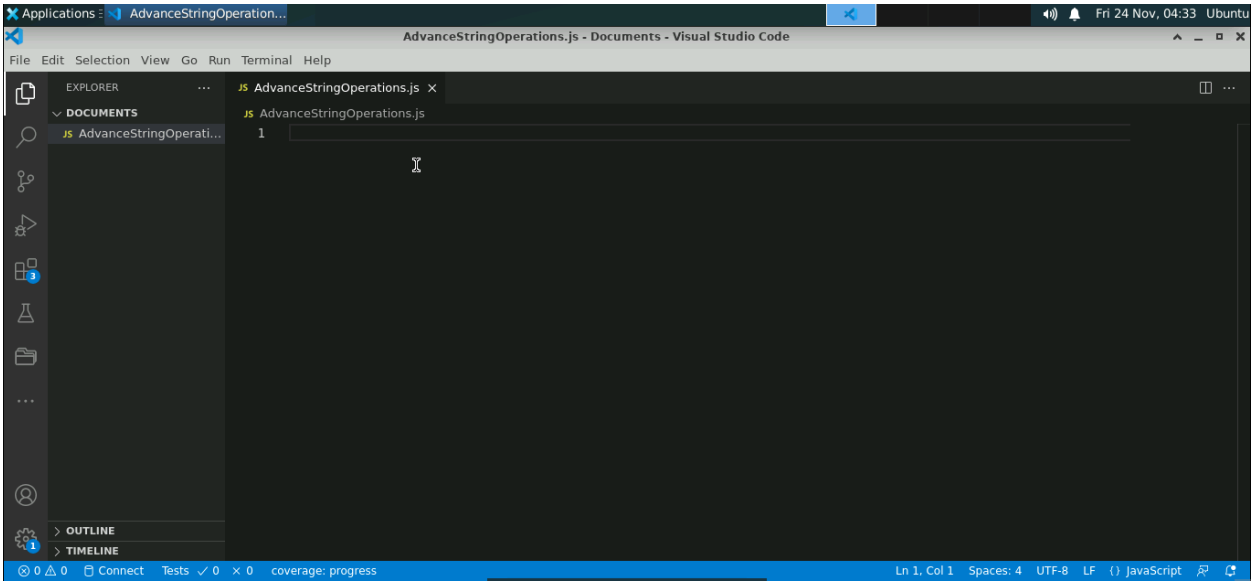
Prerequisites: A basic understanding of string operations in JavaScript

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 **AdvanceStringOperations.js**

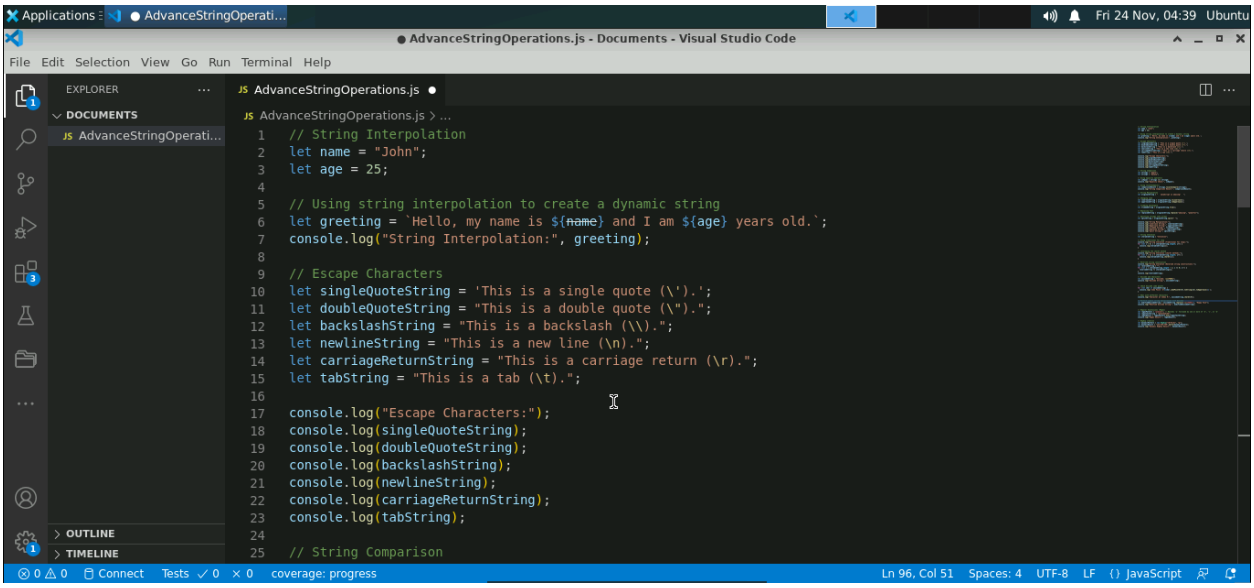


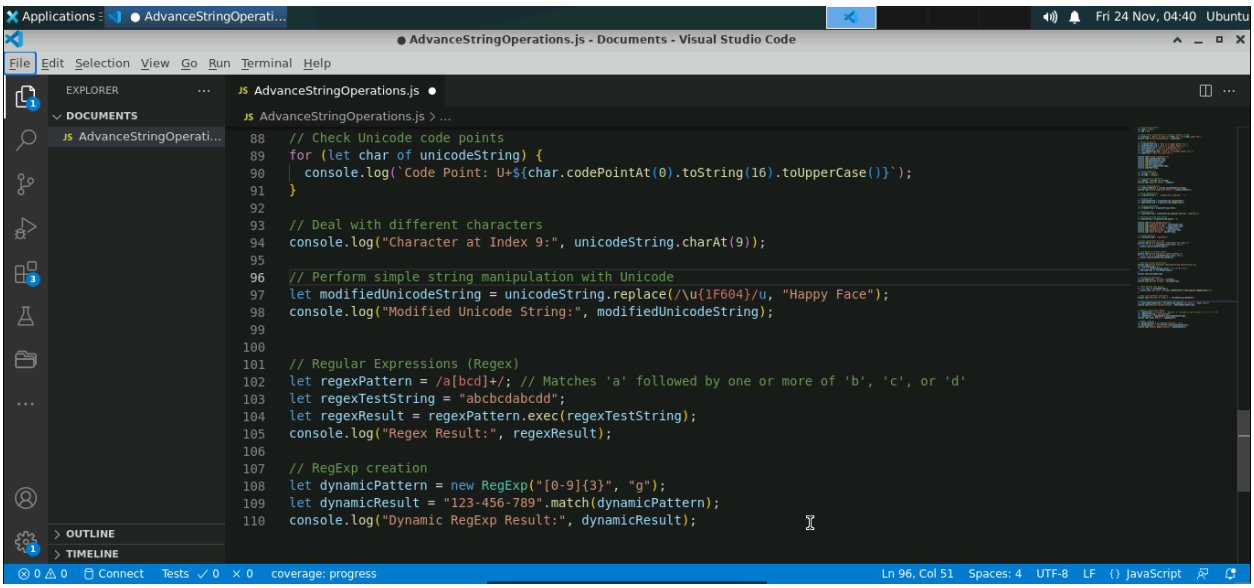
1.2 Add the following code to the **AdvanceStringOperations.js** file:

```
// String Interpolation
let name = "John";
let age = 25;

// Using string interpolation to create a dynamic string
let greeting = `Hello, my name is ${name} and I am ${age} years old.`;
console.log("String Interpolation:", greeting);

// Escape Characters
let singleQuoteString = 'This is a single quote (\').';
let doubleQuoteString = "This is a double quote (\").";
let backslashString = "This is a backslash (\\).";
let newlineString = "This is a new line (\n).";
let carriageReturnString = "This is a carriage return (\r).";
let tabString = "This is a tab (\t).";

console.log("Escape Characters:");
console.log(singleQuoteString);
console.log(doubleQuoteString);
console.log(backslashString);
console.log(newlineString);
console.log(carriageReturnString);
console.log(tabString);

// String Comparison
let string1 = "apple";
let string2 = "banana";

// Using equality operators
let isEqual = string1 === string2;
console.log("Equality Check:", isEqual);

// Using localeCompare()
let comparisonResult = string1.localeCompare(string2);
console.log("String Comparison Result:", comparisonResult);

// String Manipulation
let originalString = "   JavaScript is amazing!   ";

// Changing case
let lowerCaseString = originalString.toLowerCase();
let upperCaseString = originalString.toUpperCase();
```

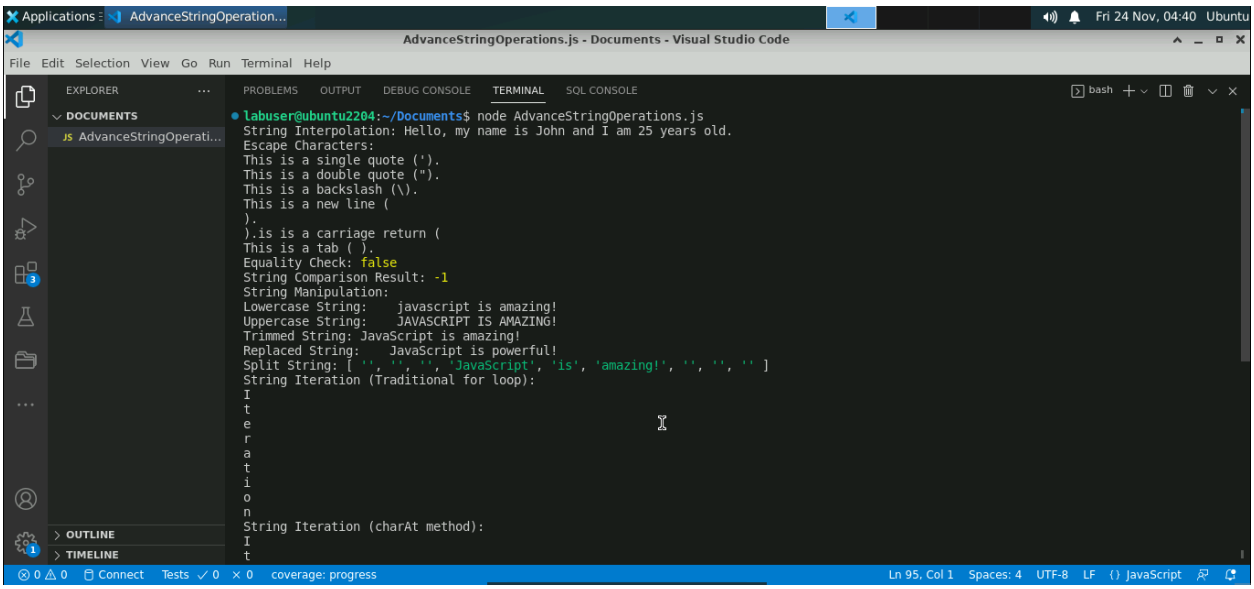
```
// Trimming whitespace
let trimmedString = originalString.trim();

// Replacing text
let replacedString = originalString.replace("amazing", "powerful");

// Splitting strings into arrays
let splitString = originalString.split(" ");

console.log("String Manipulation:");
console.log("Lowercase String:", lowerCaseString);
console.log("Uppercase String:", upperCaseString);
console.log("Trimmed String:", trimmedString);
console.log("Replaced String:", replacedString);
console.log("Split String:", splitString);

// String Iteration
let iterableString = "Iteration";

// Using traditional for loop
console.log("String Iteration (Traditional for loop):");
for (let i = 0; i < iterableString.length; i++) {
  console.log(iterableString[i]);
}

// Leveraging the charAt method
console.log("String Iteration (charAt method):");
for (let i = 0; i < iterableString.length; i++) {
  console.log(iterableString.charAt(i));
}

// Reversed string construction
console.log("String Iteration (Reversed string construction):");
let reversedString = "";
for (let i = iterableString.length - 1; i >= 0; i--) {
  reversedString += iterableString[i];
}
console.log(reversedString);

// Unicode and Characters
let unicodeString = "Unicode: \u{1F604}";
console.log("Unicode String:", unicodeString);
```

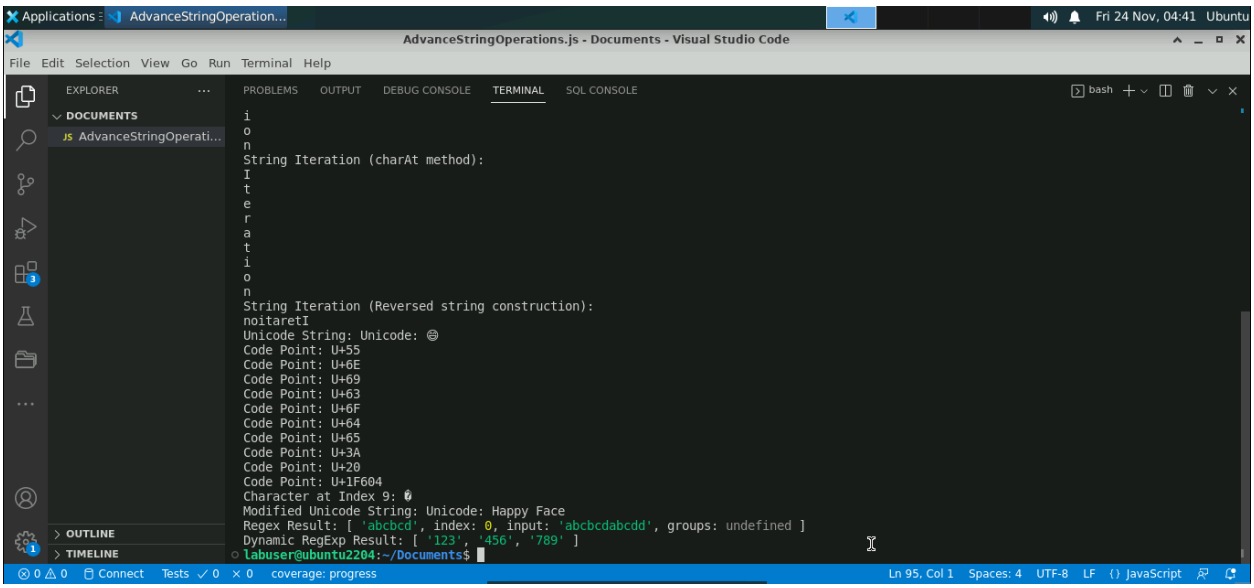
```
// Check Unicode code points
for (let char of unicodeString) {
  console.log(`Code Point: U+${char.codePointAt(0).toString(16).toUpperCase()}`);
}

// Deal with different characters
console.log("Character at Index 9:", unicodeString.charAt(9));

// Perform simple string manipulation with Unicode
let modifiedUnicodeString = unicodeString.replace(/\u{1F604}/u, "Happy Face");
console.log("Modified Unicode String:", modifiedUnicodeString);


// Regular Expressions (Regex)
let regexPattern = /a[bcd]+/; // Matches 'a' followed by one or more of 'b', 'c', or 'd'
let regexTestString = "abcbcdabcdd";
let regexResult = regexPattern.exec(regexTestString);
console.log("Regex Result:", regexResult);

// RegExp creation
let dynamicPattern = new RegExp("[0-9]{3}", "g");
let dynamicResult = "123-456-789".match(dynamicPattern);
console.log("Dynamic RegExp Result:", dynamicResult);
```





1.3 Save the file and run it using Node.js in the terminal:
node AdvanceStringOperations.js





The code demonstrates advanced JavaScript string operations, including interpolation, escape characters, and manipulation techniques. It also explores string iteration, Unicode handling, and regular expressions for pattern matching, showcasing diverse string-related functionalities.

By following these steps, you have successfully demonstrated advanced JavaScript string operations to emphasize enhanced manipulation and iteration techniques.