

Lesson 04 Demo 02

Implementing Advanced String Operations

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

Tools required: Visual Studio Code and Node.js

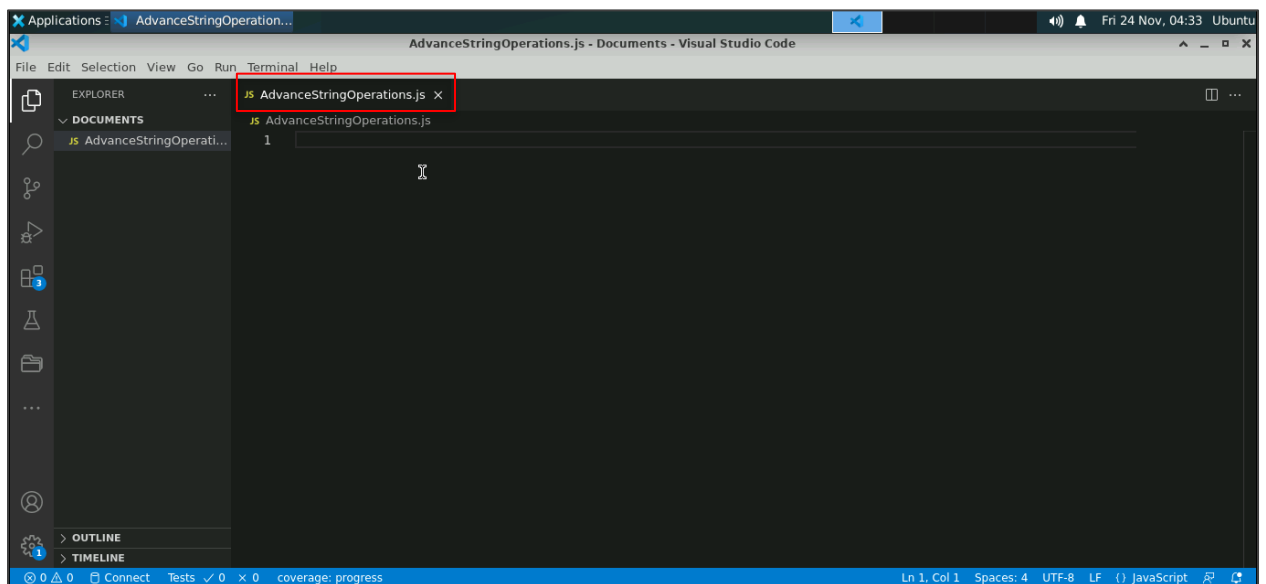
Prerequisites: A basic understanding of string operations in JavaScript

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);
```

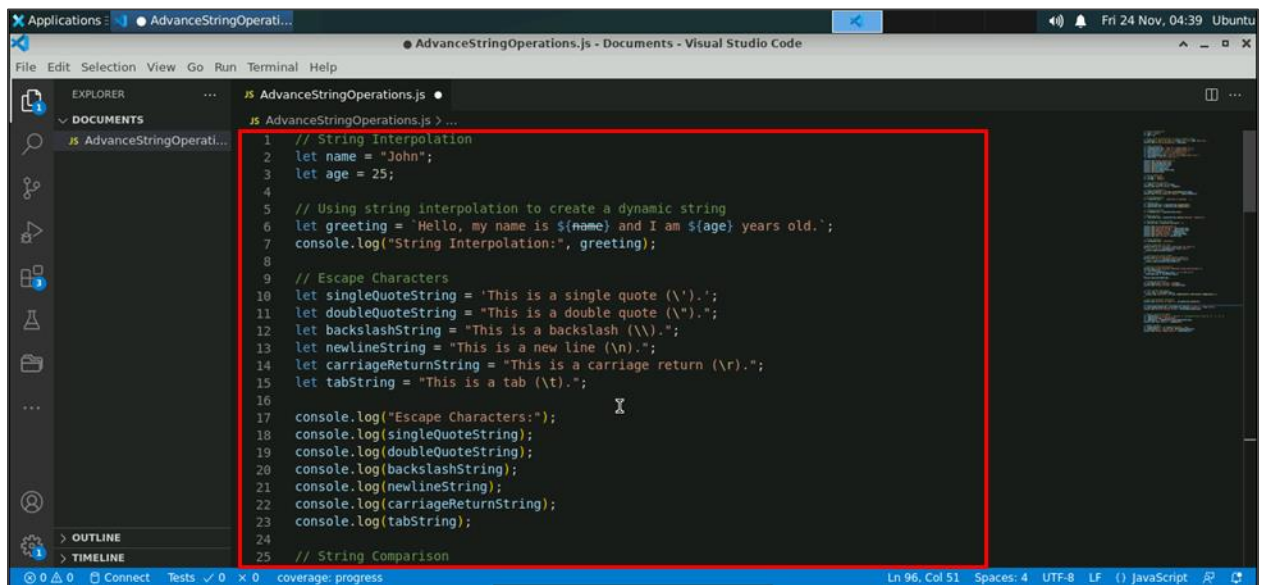
```
// Check Unicode code points
for (let char of unicodeString) {
  console.log(`Code Point: U+${char.codePointAt(0).toString(16).toUpperCase()}`);
}
```

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

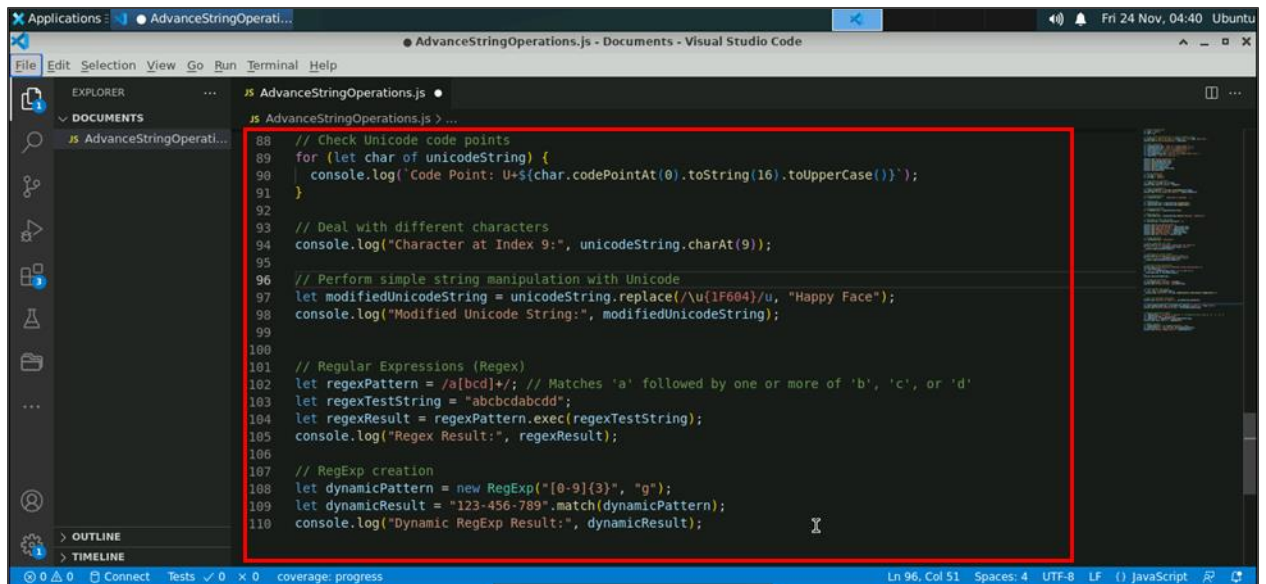
```
// Perform simple string manipulation with Unicode
let modifiedUnicodeString = unicodeString.replace(/\u{1F604}/u, "Happy Face");
console.log("Modified Unicode String:", modifiedUnicodeString);
```

```
// Regular Expressions (Regex)
let regexPattern = /a[bcd]+/; // Matches 'a' followed by one or more of 'b', 'c', or 'd'
let regexTestString = "abcbcdabccd";
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);
```

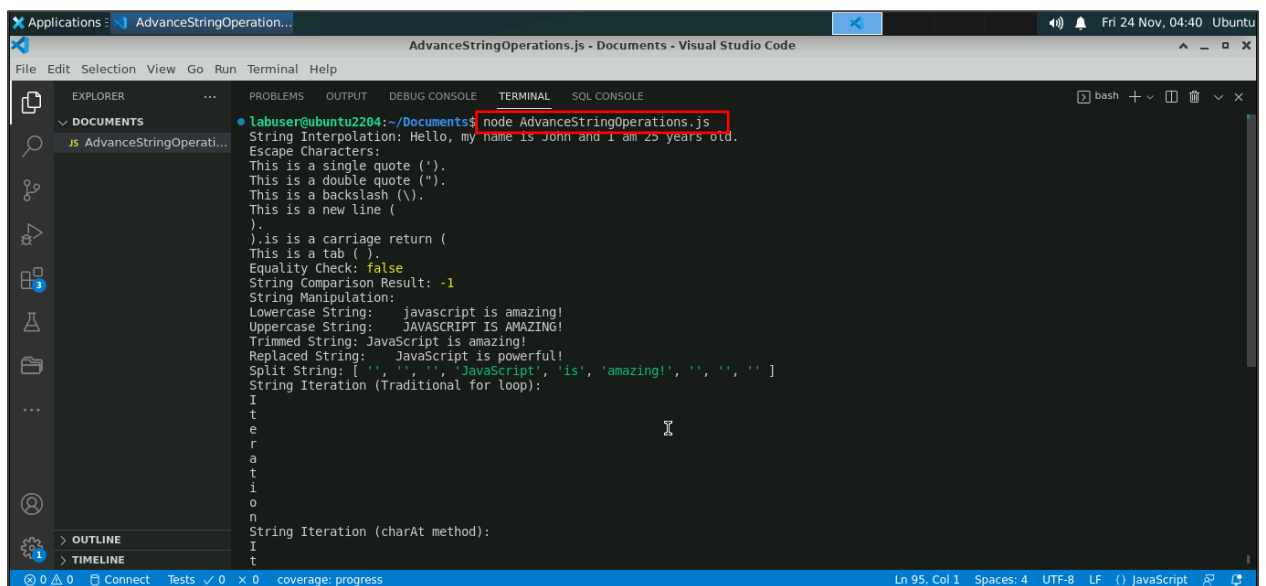


```
1 // String Interpolation
2 let name = "John";
3 let age = 25;
4
5 // Using string interpolation to create a dynamic string
6 let greeting = `Hello, my name is ${name} and I am ${age} years old.`;
7 console.log("String Interpolation:", greeting);
8
9 // Escape Characters
10 let singleQuoteString = 'This is a single quote (\').';
11 let doubleQuoteString = "This is a double quote (\").";
12 let backslashString = "This is a backslash (\\).";
13 let newlineString = "This is a new line (\n).";
14 let carriageReturnString = "This is a carriage return (\r).";
15 let tabString = "This is a tab (\t).";
16
17 console.log("Escape Characters:");
18 console.log(singleQuoteString);
19 console.log(doubleQuoteString);
20 console.log(backslashString);
21 console.log(newlineString);
22 console.log(carriageReturnString);
23 console.log(tabString);
24
25 // String Comparison
```

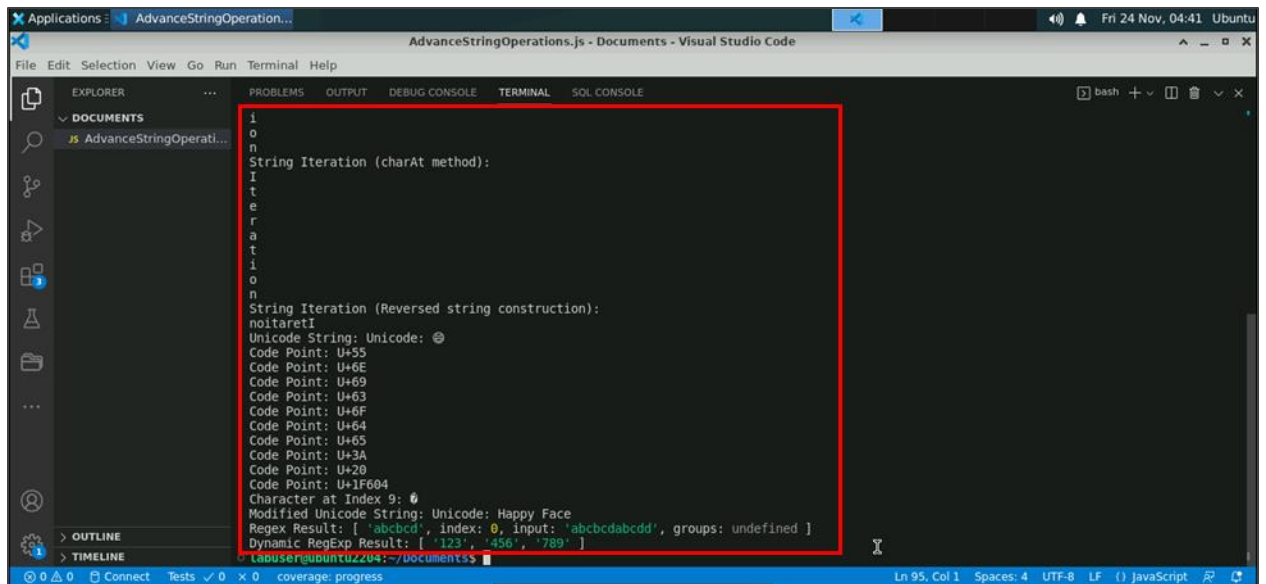


```
88 // Check Unicode code points
89 for (let char of unicodeString) {
90   console.log('Code Point: U+${char.codePointAt(0).toString(16).toUpperCase()}');
91 }
92
93 // Deal with different characters
94 console.log("Character at Index 9:", unicodeString.charAt(9));
95
96 // Perform simple string manipulation with Unicode
97 let modifiedUnicodeString = unicodeString.replace(/u{1F604}/u, "Happy Face");
98 console.log("Modified Unicode String:", modifiedUnicodeString);
99
100
101 // Regular Expressions (Regex)
102 let regexPattern = /[bcd]+/; // Matches 'a' followed by one or more of 'b', 'c', or 'd'
103 let regexTestString = "abcbcdabcbdd";
104 let regexResult = regexPattern.exec(regexTestString);
105 console.log("Regex Result:", regexResult);
106
107 // RegExp creation
108 let dynamicPattern = new RegExp("[0-9]{3}", "g");
109 let dynamicResult = "123-456-789".match(dynamicPattern);
110 console.log("Dynamic RegExp Result:", dynamicResult);
```

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



```
labuser@ubuntu2204:~/Documents$ node AdvanceStringOperations.js
String Interpolation: Hello, my name is John and I am 25 years old.
Escape Characters:
This is a single quote (').
This is a double quote (").
This is a backslash (\).
This is a new line (
).
).is is a carriage return (
This is a tab ( ).
Equality Check: false
String Comparison Result: -1
String Manipulation:
Lowercase String:  javascript is amazing!
Uppercase String:  JAVASCRIPT IS AMAZING!
Trimmed String: JavaScript is amazing!
Replaced String:  JavaScript is powerful!
Split String: [ ' ', ' ', ' ', 'JavaScript', 'is', 'amazing!', ' ', ' ', ' ' ]
String Iteration (Traditional for loop):
I
t
e
r
a
t
i
o
n
String Iteration (charAt method):
I
t
```

A screenshot of the Visual Studio Code editor interface. The main editor window displays a JavaScript file named 'AdvanceStringOperations.js'. The code is as follows:

```
i
o
n
String Iteration (charAt method):
I
t
e
r
a
t
i
o
n

String Iteration (Reversed string construction):
noitareti
Unicode String: Unicode: ☺
Code Point: U+55
Code Point: U+6E
Code Point: U+69
Code Point: U+63
Code Point: U+6F
Code Point: U+64
Code Point: U+65
Code Point: U+3A
Code Point: U+20
Code Point: U+1F604
Character at Index 0: ☺
Modified Unicode String: Unicode: Happy Face
Regex Result: [ 'abcbcd', index: 0, input: 'abcbcdabcbdd', groups: undefined ]
Dynamic RegExp Result: [ '123', '456', '789' ]
```

The code is enclosed in a red rectangular box. The editor's interface includes a sidebar on the left with 'EXPLORER', 'DOCUMENTS', and 'OUTLINE' tabs. The top status bar shows 'Ln 95, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'JavaScript'. The bottom status bar shows 'coverage: progress'.

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.