Notes on String Manipulation in JavaScript

Overview

This document covers techniques for measuring the length of strings and extracting parts of strings in JavaScript, focusing on normalizing user input (e.g., capitalizing city names) and performing string operations like checking for double spaces.

Key Points

1. **Objective**:
   - Normalize a city name (e.g., "boston", "BOSTON", "bosTon") to a standard format (e.g., "Boston").
   - Use string methods to extract parts of a string and manipulate case.
   - Measure string length and check for specific patterns (e.g., double spaces).
2. **String Indexing**:
   - Strings are indexed like arrays, with each character assigned an index starting from 0.
   - Example: For "Boston", 'B' is at index 0, 'o' at index 1, etc.
3. **The `slice` Method**:
   - Extracts a portion of a string using `slice(startIndex, endIndex)`.
   - `startIndex`: The index of the first character to include.
   - `endIndex`: The index of the first character *not* included (i.e., the slice ends at `endIndex - 1`).
   - Length of slice = `endIndex - startIndex`.
   - If `endIndex` is omitted, the slice includes all characters to the end of the string.
   - Examples:
     - `cityToCheck = "Boston";`
     - `cityToCheck.slice(0, 1)` → "B" (first character).
     - `cityToCheck.slice(2, 5)` → "sto" (characters from index 2 to 4).
     - `cityToCheck.slice(2)` → "ston" (from index 2 to end).
4. **Capitalizing a City Name**:

- Split the string into the first character and the rest, then apply `toUpperCase()` and `toLowerCase()` respectively.
- Example code:

```
var cityToCheck = prompt("Enter a city");
var firstChar = cityToCheck.slice(0, 1);
var otherChars = cityToCheck.slice(1);
firstChar = firstChar.toUpperCase();
otherChars = otherChars.toLowerCase();
var cappedCity = firstChar + otherChars;
```

- Steps:
  (a) Extract first character (`firstChar`).
  (b) Extract remaining characters (`otherChars`).
  (c) Capitalize `firstChar`.
  (d) Lowercase `otherChars`.
  (e) Concatenate to form `cappedCity` (e.g., "Boston").
- Note: The original code has a typo ("otherChars publication" should be "otherChars.toLowerCase()").

5. **Measuring String Length**:

- Use the `length` property to count characters in a string.
- Example:

```
var month = prompt("Enter a month");
var charsInMonth = month.length;
if (charsInMonth > 3) {
    var monthAbbrev = month.slice(0, 3);
}
```

- If `month = "November"`, `charsInMonth = 8`, and `monthAbbrev = "Nov"`.

6. **Checking for Double Spaces**:

- Loop through a string to check for consecutive spaces using `slice` and the `length` property.
- Example code:

```
var str = prompt("Enter some text");
var numChars = str.length;
for (var i = 0; i < numChars; i++) {
    if (str.slice(i, i + 2) === "  ") {
        alert("No double spaces!");
        break;
    }
}
```

- Steps:
  (a) Get string length (`numChars`).

(b) Loop through the string, checking each 2-character segment.

(c) If a double space is found, display an alert and exit the loop.

- Note: The alert message "No double spaces!" is misleading; it should indicate that double spaces *were* found. Also, "1" in the loop should be "i".

7. **Additional Notes**:

   - String `length` is 1-based, but indexing is 0-based, so loops typically run until `i < length`.
   - The `slice` method does not modify the original string; it returns a new string.
   - The original document assumes single-word city names (e.g., ignores "New Orleans").

Observations

- **Typos in Original Code**:
  - "otherChars publication" should be "otherChars.toLowerCase()".
  - "toUperCaseU" (from previous document) should be "toUpperCase".
  - In the double-space check, "1" should be "i" in the loop.
  - The alert message in the double-space check is incorrect.
- **Loop Issue**:
  - The loop in the double-space check should use `i < numChars - 1` to avoid out-of-bounds errors when checking `i + 2`.
- **Improvements**:
  - The double-space check could use `str.includes(" ")` for simplicity in modern JavaScript.