
◆ Problem: Formatting User Input Properly

When users enter city names, they may type them in various ways:

- `"boston"` (all lowercase)
- `"BOSTON"` (all uppercase)
- `"bosTon"` (mixed case)

To maintain **consistent formatting**, we want to convert the input into a **properly capitalized format**, like `"Boston"`.

⊘ Why `.toLowerCase()` or `.toUpperCase()` Alone Won't Work

- `.toLowerCase()` converts **everything** to lowercase → `"boston"`.
- `.toUpperCase()` converts **everything** to uppercase → `"BOSTON"`.
- But we need `"Boston"`, where only the **first letter is uppercase** and the rest are lowercase.

◆ Solution: Using `slice()` to Separate and Reformat the String

To format the city name correctly:

1. Extract the **first letter**.
2. Extract the **remaining characters**.
3. Convert the **first letter to uppercase**.
4. Convert the **remaining characters to lowercase**.
5. **Combine them back** into a single string.

◆ Code Explanation

```
var cityToCheck = prompt("Enter your city"); // Get user input
var firstChar = cityToCheck.slice(0, 1); // Extract first character
var otherChars = cityToCheck.slice(1); // Extract remaining characters

firstChar = firstChar.toUpperCase(); // Convert first character to uppercase
otherChars = otherChars.toLowerCase(); // Convert remaining characters to lowercase

var cappedCity = firstChar + otherChars; // Combine the two parts
console.log(cappedCity); // Output the formatted city name
```

◆ Step-by-Step Breakdown

Line	Code	Explanation
1	var cityToCheck = prompt("Enter your city");	Ask the user for a city name.
2	var firstChar = cityToCheck.slice(0, 1);	Extract **only the first character**.
3	var otherChars = cityToCheck.slice(1);	Extract **everything else** after the first character.
4	firstChar = firstChar.toUpperCase();	Convert the **first letter** to uppercase.
5	otherChars = otherChars.toLowerCase();	Convert the **rest** of the string to lowercase.
6	var cappedCity = firstChar + otherChars;	Concatenate both parts into a correctly formatted city name.

◆ Understanding the `slice()` Method

The `.slice(start, end)` method **extracts a portion** of a string.

Example 1: Extracting the First Letter

```
var cityToCheck = "Boston";
var firstChar = cityToCheck.slice(0, 1); // "B"
```

- Starts at **index `0`** (the first letter).
- Stops at **index `1`**, so it only captures `"B"`.

Example 2: Extracting the Rest of the String

```
var otherChars = cityToCheck.slice(1); // "oston"
```

- Starts at **index `1`** (the second letter).
- **No second parameter** → extracts the rest of the string.

◆ Using `length` to Check String Size

The `.length` property **counts the number of characters** in a string.

Example: Trimming a Month Name

```
var month = prompt("Enter a month");
var charsInMonth = month.length;

if (charsInMonth > 3) {
  monthAbbrev = month.slice(0, 3); // Extract first 3 characters
}

console.log(monthAbbrev);
```

- If the month name is "November", the output is "Nov".
- This method is useful for shortening long strings.

◆ Looping Through a String to Detect Double Spaces

We can **iterate through a string** using a loop to check for double spaces " ".

◆ Code Example

```
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; // Stop the loop once double spaces are found
    }
}
```

◆ How It Works

1. **Gets user input**.
2. **Counts the number of characters** in the string using .length.
3. **Loops through the string**, checking each **2-character slice** (slice(i, i+2)).
4. If a **double space** is found, it **alerts the user and stops**.

◆ Summary of Key Concepts

Concept	Explanation
Case Sensitivity	JavaScript treats "Boston" and "boston" as different strings.
.slice(start, end)	Extracts a substring from a string.
First Letter Capitalization	Extract the first letter, capitalize it, and lowercase the rest.
.length	Counts the number of characters in a string.
Looping Through a String	Used to check for patterns like double spaces.