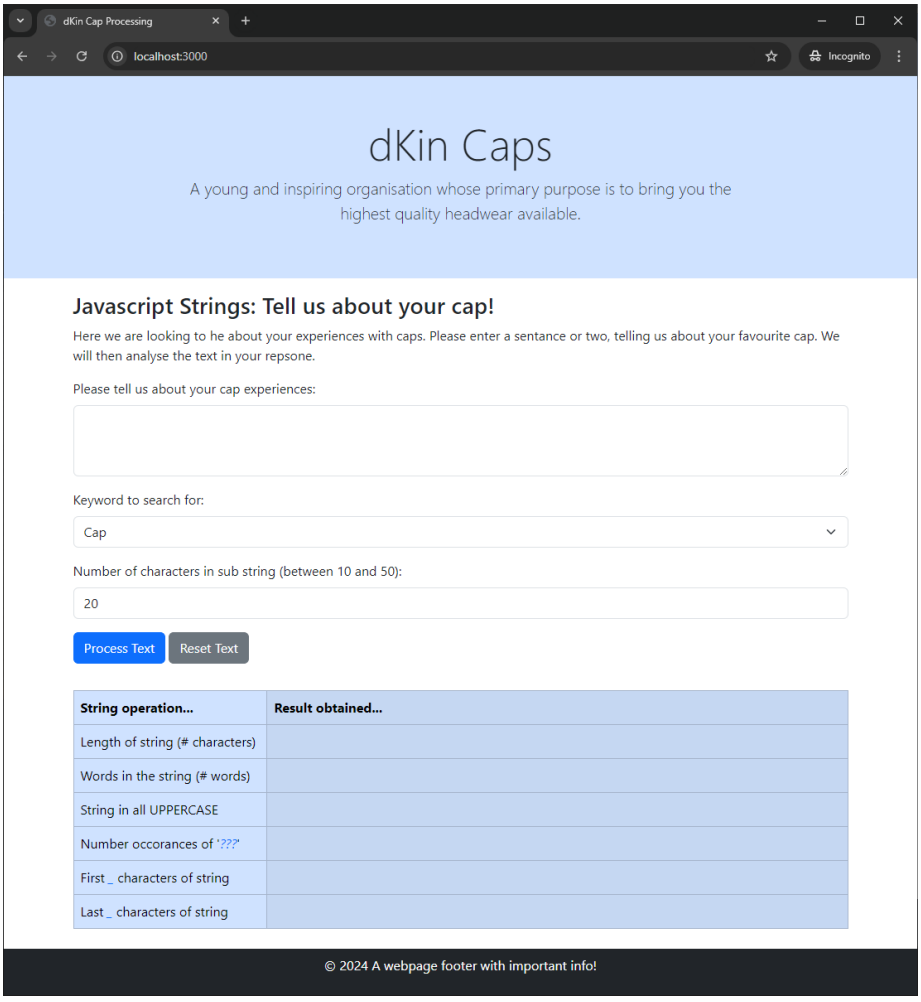
7.1P: String Objects

Tasks

In this task you will be exploring the Javascript built-in *String Object*, along with some of its common methods for doing useful work.

To complete this task, you are asked to use the provided simple form that obtains from the user three inputs (an input string, a string to search for, and the number of characters setting the size of a substring). Your code should then, based on this input, provide the output from various *string object* operations. The actual layout/design of the web page is left up to you, but you can use the sample provided in the supplied code, which displays as shown below:



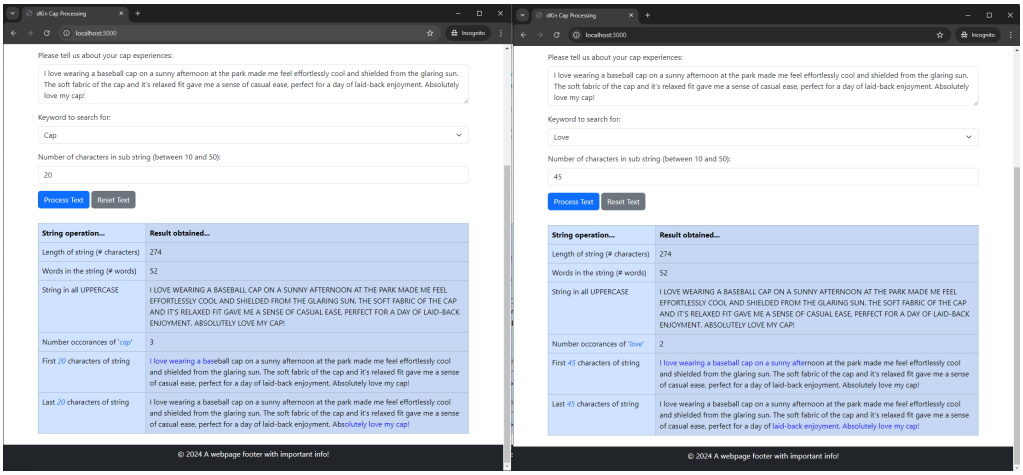
Task7.1.1 Input Form and Blank Results

The image below shows the output given with an input string:

```
`I love wearing a baseball cap on a sunny afternoon at the park made me feel effortlessly cool and shielded from the glaring sun. The soft fabric of the cap and it's relaxed fit gave me a sense of casual ease, perfect for a day of laid-back enjoyment. Absolutely love my cap!`
```

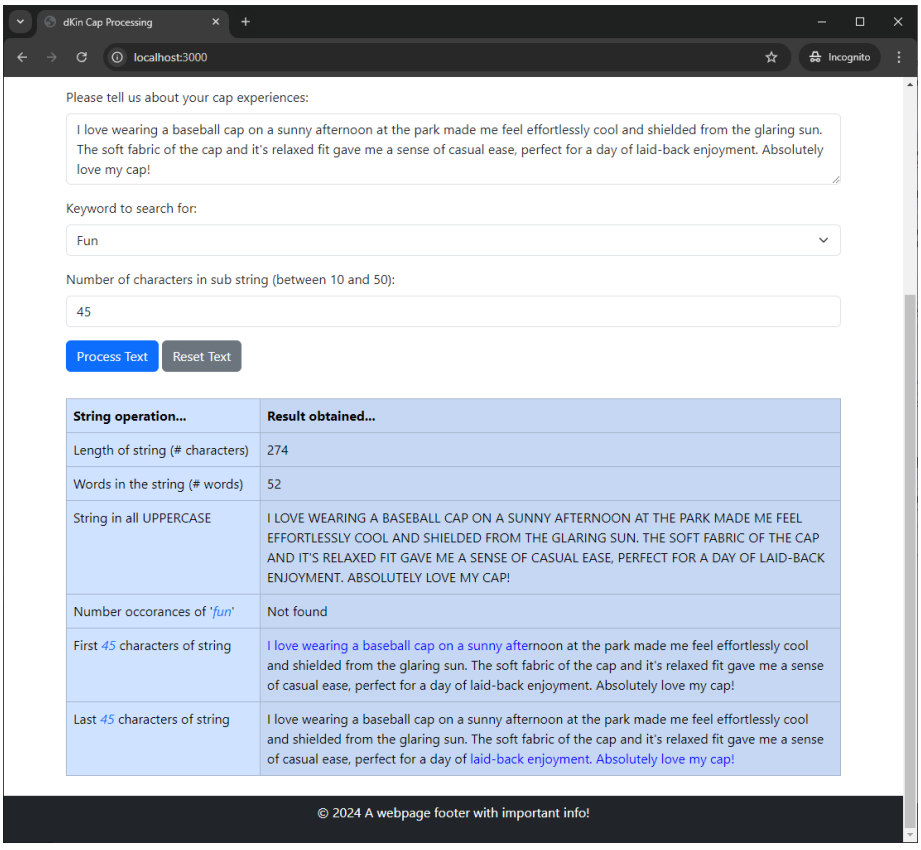
Followed by the search string obtained from a selection input, and finally a number

input form a range between 10 and 50 to be used as the size of the substrings (one from the start of the input string, the other form the end of the input string). In the *left* example, the search string is `Cap` and the substring size is `20`, while the *right* example has a search string of `Love` and a substring size of `45`.



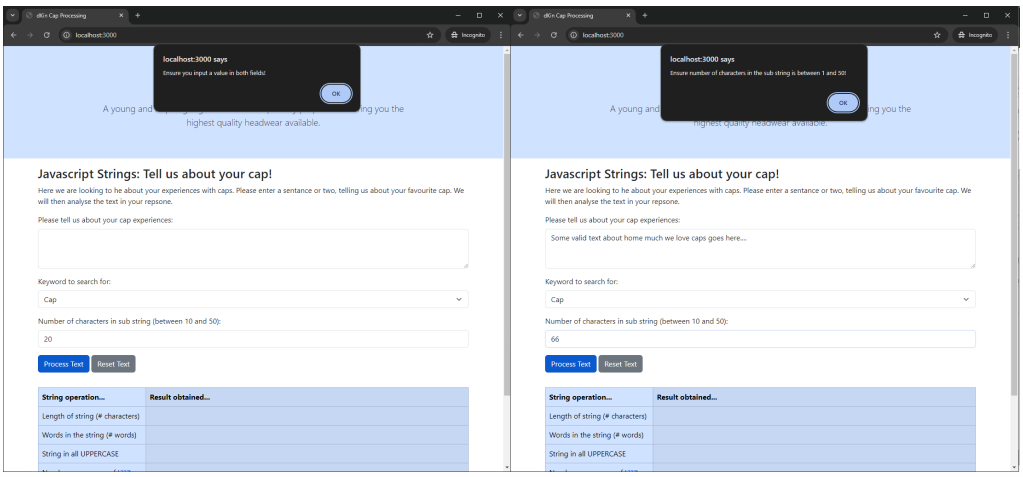
Task7.1.2 Data input and results

If the search string provided does not exist in the input string then a value of `Not Found` should be displayed, as shown below:



Task7.1.3 Data input and no search string found

Additionally, if **no** input string or invalid number of characters in the substring is provided, relevant **alert** messages should be displayed, as shown in the screenshots below:



Task7.1.4 Invalid input data provided

Finally, if the `Reset Text` button is pressed, the input and result fields should all be cleared to match the initial state of the page.

Steps

The provided web page has the respective **input** and **output** fields individually identified using the HTML `id` attribute. These can be used to obtain the values for processing, as well as for updating the table.

In example provided, the results fields start off as being empty and then when the user clicks the `Process Text` button, the respective fields are updated with the output from the various *string methods*.

Each of the input string variables will be stored as a javascript `String` inbuilt object that comes with a number of helpful/useful methods (functions) to perform work.

The specific *string* operations to perform on the *input string* include:

1. Display the total **number of characters** from the provided *input string*.
2. Display the total **number of words** found in the provided *input string*. > NOTE: Consider using the `String.split(" ")` method here to create an array of strings (words) from the input. Then you can display the size (`length`) of the string.
3. Display all characters of the *input string* in **UPPERCASE**.
4. Display the **number of occurrences** of the *input substring* found within the *input string* (should display `Not found` if it doesn't exist in the input string)

   > NOTE: Convert the search string into a *lowercase* and then search in the *lowercase input string* to be case **insensitive** (There are other ways to do this and you are welcome to implement these if you like!)
5. Display and highlight in a different colour the **first `n` characters** of the *input string*, where `n` is provided by the user. > NOTE: Create the result string in two parts, the first *n* characters, which is wrapped up in a `<span style="blue">`

`... </span>` styling. The remaining n+1 to stringlength is plain text.

6. Display and highlight the the **last `n` characters** of the *input string*, where `n` is provided by the user.

Hints

As done in the tasks from Week 06, create a javascript file that holds **two** functions that are called on the event one of the buttons is clicked (e.g., one for `processText()` and the other for `clearText()`). You should be able to use the unique IDs to the respective input and output fields within the *Input Data* form and *Processed Results* section, to extract/update these fields from the javascript code.

The `alert` error message is displayed when either the *input string* **OR** the *search string* is empty, e.g.:

```
if( inputString == "" || searchString == "" ) {
    ... alert error message ...
} else if( numCharacters < 10 || numCharacters > 50 ) {
    ... alert error message ...
} else {
    ... process the form input and
    ... display the results
}
```

Here, the `numCharacters` taken from the form will be by default a `string` value. To convert it to a number use the JS method `parseInt(inputstring)`.

Furthermore, review the various methods/functions available to work with *JavaScript String Objects*:

- https://www.w3schools.com/js/js_string_methods.asp
- https://www.w3schools.com/jsref/jsref_obj_string.asp

References for JS pop-up boxes, including `alert()`:

- https://www.w3schools.com/js/js_popup.asp

What will you submit?

You should submit:

- Screen-shot of the web page with valid input, search string and substring length.
- Screen-shot of the web page with valid input, a different search string and different substring length.
- Screen-shot of the web page with valid input and non-existent substring.
- Screen-shot of the web page missing the input and showing the `alert` error message.
- Your HTML file.
- Your JavaScript file.