

# Lab 9

**Due** Apr 16, 2023 by 11:59pm


**Points** 100


**Submitting** a file upload

## CS-546 Lab 9

### Client-Side Text Analyzer

For this lab, you will be using HTML, CSS, and JavaScript on the user's browser to make a simple client-side text analyzer!

You will create an express server with a single page at the location  that will provide the user with a web page to allow them to enter some text, it will then calculate various statistics based on that text. **The entire checking operation will be done using client-side JavaScript. Major points will be deducted if you perform the processing server-side.**

YOU MUST use the directory and file structure in the code stub or points will be deducted. You can download the starter template here: [Lab9\\_stub.zip \(https://sit.instructure.com/courses/64637/files/11413569?wrap=1\)](https://sit.instructure.com/courses/64637/files/11413569?wrap=1).   
([https://sit.instructure.com/courses/64637/files/11413569/download?download\\_frd=1](https://sit.instructure.com/courses/64637/files/11413569/download?download_frd=1))

**PLEASE NOTE: THE STUB DOES NOT INCLUDE THE PACKAGE.JSON FILE. YOU WILL NEED TO CREATE IT! DO NOT ADD ANY OTHER FILE OR FOLDER APART FROM PACKAGE.JSON FILE.**

## The Server

**Your server this week should not be doing any of the processing! Your server only exists to allow someone to get to the HTML Page and download the associated assets to run the Text Analyzer Page.**

### The Whole Text Analyzer Application.

Your page should have a few basic user interface elements:

- A header tag, with an h1 naming your site, with a title for your page.
- A footer with your name, student ID, and any other info about yourself you wish to include
- A single div with an id of `results`.

Your page will have a form with the following:

- A label with a `for` attribute referencing your input
- A textarea element with a `name` and `id` of `text_input`
- A button to submit the form

Using JavaScript in your browser only, you will listen for the form's `submit` event; when the form is submitted, you will:

- Get the value of the input text element.
  - You will take in the text input , convert it to all lowercase and generate some text statistics based on the input.
  - You will calculate the following statistics based on the text:
    - Original Input: you will just show the input that the user entered (see below)
    - Total Letters: total number of letter characters in the text ,
    - Total Non-Letters: total number of non-letters in the text (including spaces),
    - Total Vowels: total number of vowels in the text (not counting y),
    - Total Consonants: total number of consonants in the text (counting y),
    - Total Words: total number of words in the text; a word is defined as any sequence of letters broken by any not-letter. For example, the phrase to-do is two words; a word does not start until a letter appears,
    - Unique Words: total number of unique words that appear in the lowercased text,
    - Long Words: number of words in the text that are 6 or more letters long; this is a total count of individual words, not unique words,
    - Short Words: number of words in the text that are 3 or less letters long; this is a total count of individual words, not unique words

This lab is easy to over-complicate by attempting to be too clever. I am giving two important pieces of advice:

- You will generate the following HTML every time the application processes the text and append it to the results div.

You will be using a data list element (dl), inside the dl, you will have a data title (dt) that has the title of the stat and then a data description (dd) which has the value. (see expected output below)

- Here is the output based on the input: "Hello, my -! This is a great day to say hello. Hello! 2 3 4 23"

```
<dl>
  <dt>Original Input:</dt>
  <dd>Hello, my -! This is a great day to say hello. Hello! 2 3 4 23</dd>
  <dt>Total Letters</dt>
  <dd>40</dd>
  <dt>Total Non-Letters</dt>
  <dd>27</dd>
  <dt>Total Vowels</dt>
  <dd>14</dd>
  <dt>Total Consonants</dt>
  <dd>26</dd>
  <dt>Total Words</dt>
  <dd>11</dd>
  <dt>Unique Words</dt>
  <dd>9</dd>
```

```

<dt>Long Words</dt>

<dd>3</dd>

<dt>Short Words</dt>

<dd>6</dd>

</dl>

```

You will generate the above HTML and append it to the div every time the form is submitted, so you will have multiple data lists (dl) in the div, one for each time the user inputs and processes some text. So for example:

If the user submitted the following input and processed it:

1. "Hello, my -! This is a great day to say hello. Hello! 2 3 4 23"
2. "The quick brown fox jumps over the lazy dog."
3. "Hello, my -! This is a great day to say hello. Hello! 2 3 4 23"

Your div would look like this:

```

<div id="results">

  <dl>

    <dt>Original Input:</dt>

    <dd>Hello, my -! This is a great day to say hello. Hello! 2 3 4 23</dd>

    <dt>Total Letters</dt>

    <dd>40</dd>

    <dt>Total Non-Letters</dt>

    <dd>27</dd>

    <dt>Total Vowels</dt>

    <dd>14</dd>

    <dt>Total Consonants</dt>

    <dd>26</dd>

    <dt>Total Words</dt>

    <dd>11</dd>

    <dt>Unique Words</dt>

    <dd>9</dd>

    <dt>Long Words</dt>

    <dd>3</dd>

    <dt>Short Words</dt>

    <dd>6</dd>

  </dl>

  <dl>

```

```
<dt>Original Input:</dt>

<dd>The quick brown fox jumps over the lazy dog.</dd>

<dt>Total Letters</dt>

<dd>35</dd>

<dt>Total Non-Letters</dt>

<dd>9</dd>

<dt>Total Vowels</dt>

<dd>11</dd>

<dt>Total Consonants</dt>

<dd>24</dd>

<dt>Total Words</dt>

<dd>9</dd>

<dt>Unique Words</dt>

<dd>8</dd>

<dt>Long Words</dt>

<dd>0</dd>

<dt>Short Words</dt>

<dd>4</dd>

</dl>

<dl>

  <dt>Original Input:</dt>

  <dd>Hello, my -! This is a great day to say hello.   Hello! 2 3 4 23</dd>

  <dt>Total Letters</dt>

  <dd>40</dd>

  <dt>Total Non-Letters</dt>

  <dd>27</dd>

  <dt>Total Vowels</dt>

  <dd>14</dd>

  <dt>Total Consonants</dt>

  <dd>26</dd>

  <dt>Total Words</dt>

  <dd>11</dd>

  <dt>Unique Words</dt>

  <dd>9</dd>

  <dt>Long Words</dt>
```

```
<dd>3</dd>

<dt>Short Words</dt>

<dd>6</dd>

</dl>

</div>
```

If the user does not have a value for the input when they submit, you should not continue processing and instead should inform them of the error on the page. If the user enters bad data, you should not continue processing and instead inform them of the error on the page.

The form should reset itself every time after an input has been processed.

## The style

You will style your page using at least 5 CSS selectors for general CSS styling. You will place the CSS in its own file.

## References and Packages

Basic CSS info can easily be referenced in the [MDN CSS tutorial \(https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting\\_started\)](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started).

## Requirements

1. All previous requirements still apply.
2. You **must remember** to update your package.json file to set `app.js` as your starting script!
3. **Your HTML must be valid** ([https://validator.w3.org/#validate\\_by\\_input](https://validator.w3.org/#validate_by_input)) or you will lose points on the assignment.
4. Your HTML must make semantical sense; usage of tags for the purpose of simply changing the style of elements (such as `i`, `b`, `font`, `center`, etc) will result in points being deducted; think in terms of content first, then style with your CSS.
5. **You can be as creative as you'd like to fulfill front-end requirements**; if an implementation is not explicitly stated, however you go about it is fine (provided the HTML is valid and semantical). Design is not a factor in this course.
6. **Your client side JavaScript must be in its own file and referenced from the HTML accordingly.**
7. All inputs must be properly labeled!