

# CS-546 Lab 8

## Palindromes

---

For this lab, you will be using HTML, CSS, and JavaScript on the user's browser to make a simple palindrome checker!

A palindrome is a phrase that is spelled the same way, backwards and forwards (ignoring spacing and punctuation). For example, the following phrases are palindromes:

- Madam
- Was it a cat I saw?
- He did, eh?
- Go hang a salami, I'm a lasagna hog.
- Poor Dan is in a droop

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 check if a phrase is a palindrome.

## The Server

---

**Your server this week should not check for palindromes! Your server only exists to allow someone to get to the HTML Page and download the associated assets to run the palindrome checking page.**

## The Page

---

Your page should have a few basic user interface elements:

- A header, with a heading, with a title for your page
- A footer with your name, student ID, and any other info about yourself you wish to include
- A list with all the terms you have checked so far; words that are palindromes will be colored in blue, while words that are not will be colored in red.

Your page will have a form with the following:

- A textarea
  - 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 textarea
  - Lowercase the text
  - Strip all non alphanumeric text; this includes spaces. For example, `Hello, 2 the world!` becomes `hello, 2 the world!` when lowercased and then `hello2theworld` when stripped of all non alphanumeric text
  - Determine whether or not the text is a palindrome
  - Add a list item to the list of terms you have checked. This list item should have a class of `is-palindrome` if it is a palindrome, or `not-palindrome` if it is not.

If the user does not have a value for the textarea when they submit, you should not continue the palindrome checking and instead should inform them of an error somehow.

# The style

---

You will style your page using at least 10 CSS selectors for general CSS styling. You will place the CSS in its own file.

You must style the `is-palindrome` class to have some sort of blue color (`#0000FF` is pure blue, you do not need to use that exact hex code), and `not-palindrome` to have some sort of red color (`#FF0000` is pure red, you do not need to use that exact hex code).

## References and Packages

---

Basic CSS info can easily be referenced in the [MDN CSS tutorial \(Links to an external site.\)](#)[Links to an external site.](#). If you need a quick CSS reference,

You will use the **express** package as your server.

You can read up on [express \(Links to an external site.\)](#)[Links to an external site.](#) on its home page. Specifically, you may find the [API Guide section on requests \(Links to an external site.\)](#)[Links to an external site.](#) useful.

You may use the [lecture 4 code \(Links to an external site.\)](#)[Links to an external site.](#) as a guide.

You may use the [lecture 5 code \(Links to an external site.\)](#)[Links to an external site.](#) as a guide.

You may use the [lecture 6 code \(Links to an external site.\)](#)[Links to an external site.](#) as a guide.

You may use the [lecture 8 code \(Links to an external site.\)](#)[Links to an external site.](#) as a guide.

## Requirements

---

1. You **must not submit** your `node_modules` folder
2. You **must remember** to save your dependencies to your `package.json` folder
3. You must do basic error checking in each function
  1. Check for arguments existing and of proper type.
  2. Throw if anything is out of bounds (ie, trying to perform an incalculable math operation or accessing data that does not exist)
  3. If a function should return a promise, instead of throwing you should return a rejected promise.
4. You **must remember** to update your `package.json` file to set `app.js` as your starting script!
5. [Your HTML must be valid \(Links to an external site.\)](#)[Links to an external site.](#) or you will lose points on the assignment.
6. 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.
7. **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.
8. **Your client side JavaScript must be in its own file and referenced from the HTML accordingly.**
9. All inputs must be properly labeled!