

# Lab 9

[New Attempt](#)

---

**Due** Nov 12 by 11:59pm    **Points** 100    **Submitting** a file upload

---


## CS-546 Lab 9

### 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 entire checking operation will be done using client-side JavaScript.**

### 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 Whole Palindrome Checker

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 ordered list with an id of `attempts`. All attempted strings with all the terms you have checked so far (until you refresh the page) will appear in this list as list items. Phrases that are

palindromes will be colored in *blue*, while phrases that are not will be colored in *red*. You must use the CSS classes below to color these phrases.

Your page will have a form with the following:

- A textarea with a `name` of `phrase`
- 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 `#attempts` 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 make text have a `color` of `#0000FF` and `not-palindrome` class to make text have a color of `#FF0000`.

## 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!