CIS 111
Introduction to Web Programming

Project 4 – Utility Hub (Part 2)

1. Overview

The **Utility Hub** is a simple web-based application designed to integrate two key utilities:

1. A **Basic Calculator** for arithmetic operations with history tracking.
2. A **NATO Phonetic Alphabet Converter** to transform text into its NATO phonetic equivalent.

The project is divided into two main sections: **Calculator Functionality** and **NATO Phonetic Converter**.

This project will require you to examine the HTML and CSS to determine which CSS/HTML classes or ids are required.

2. Incremental Development

Incremental development means building part of a program before making it all. It is a staging strategy in which product features are developed and delivered to users at different times, increasing its functionality and complexity.

Complete each of the following project stages.

2.1 The starter code

The starter code includes the following files in the Project4 folder.

- p4.html
- p4.css
- p4.js

Important: *You should finish Part 1 before you start with Part 2. You should examine the starter code for required functions and elements. Before adding any code to the body of any functions, you should use single-line comments to map out the steps.*

NATO Phonetic Converter

Note that the starter code includes adding a callback function (**setup**) to the browser **window** for the **DOMContentLoaded** event.
When attempting to add callback functions as soon as the web page loads, you must wait until the DOM is completely loaded; otherwise, attempting to reference unloaded HTML elements will result in return a **null** value and the code will fail.
The **DOMContentLoaded** event is the modern event used to wait for the DOM to load.

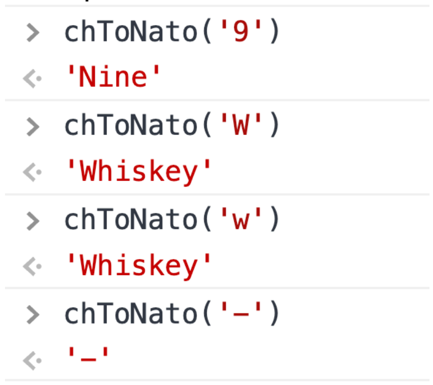
Remember that the **setup()** function adds callback functions for the button **click** events, and you need to add more addEventListeners for the **verbalize(…)** and **clearNATOInputs()** functions.

2.2 [10 pts] The chToNato() Function

Write a function chToNato() that accepts an alphanumeric character (uppercase or lowercase) and returns the NATO Phonetic version of the character. For all other symbols, return the original character. You must use a JSON object to store the letters, digits, and their NATO phonetic equivalent. Below is an example code.

(TIP: JSON objects have quotation marks around object property names).

Examples:



```
> chToNato('9')
<· 'Nine'
> chToNato('W')
<· 'Whiskey'
> chToNato('w')
<· 'Whiskey'
> chToNato('-')
<· '-'
```

1. Create a JSON object using the following code format example. This is an example.

   ```
   const text = ‘{“school”:”UO”, “city”:”Eugene”, “state”:”Oregon”}’;
   ```

   To access a JSON object using the following code format example. Use the name to access the value. This is an example.

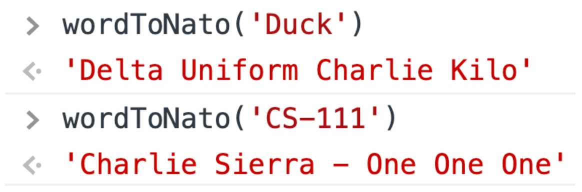
   ```
   text[“school”];
   ```

2. Return the value of character if it exists in the JSON object, else return the input parameter. (If you choose to only have one set of alphabets, say either in upper case or lower case for the name, user an appropriate case converter for the input parameter to use as a lookup for the JSON object name.)

2.3 [10 pts] The wordToNato() Function: Getting the Numbers

Write a function wordToNato(word) that accepts a word and returns a string consisting of the NATO phonetic for each letter. The wordToNato() function must call the chToNato() function. Use a space between each NATO phonetic equivalent.

Examples:



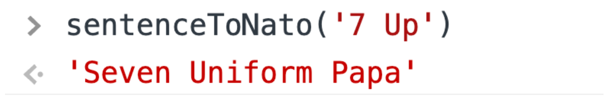
```
> wordToNato('Duck')
<· 'Delta Uniform Charlie Kilo'
> wordToNato('CS-111')
<· 'Charlie Sierra - One One One'
```

1. Use **.split(““)** to split the characters of the word.
2. Use **.map()** and pass the words to chToNato function.
3. Use **.join(“ “)** to concatenate the results with space in between.

2.4 [10 pts] The sentenceToNato() Function

Write a function sentenceToNato(sentence) that accepts a word or phrase and returns a string consisting of the NATO phonetic for each word in the phrase. The sentenceToNato() function must call your wordToNato() function.

Examples:



```
> sentenceToNato('7 Up')
<· 'Seven Uniform Papa'
```

1. Use **.split(“ “)** to split the words of the sentence using the space character.
2. Use **.map()** and pass the words to wordToNato function.
3. Use **.join(“ “)** to concatenate the results with space in between.

2.5 [10 pts] The verbalize() Function

Write a function that displays the NATO phonetic representation of an input string.

Example: verbalize() updates the HTML to display the NATO version of the input.

1. Use **document.getElementById().value** to get the input string.
2. Pass in the input string to **sentenceToNato()** to get the result.

3. Display the result to natoResult element by assigning the result to **document.getElementById().textContent.**
4. Save and test the code and confirm the calculated result displays in the **result input** box.

2.6 [10 pts] The clearNATOInputs() Functions

1. Use **querySelector()** and the ids: inputString and natoResult.

2.7 Submission

Upload p4.js, p4.html and p4.css files to Canvas.

Remember, you are responsible to submit your projects before the deadline. Late submissions will not be accepted.