JavaScript Unique Morse Code Words

Challenge

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows:

- `'a'` maps to `'.-'`,
- `'b'` maps to `'-...'`,
- `'c'` maps to `'-.-.'`, and so on.

For convenience, the full table for the `26` letters of the English alphabet is provided below:

```
['.-','-...','-.-.','-..','.','..-.','--.',
'....','..','.---','-.-','.-..','--','-.',
'---','.--.','--.-','.-.','...','-','..-',
'...-','.--','-..-','-.--','--..']
```

Given an array of strings `words` where each word can be written as a concatenation of the Morse code of each letter.

For example, `'cab'` can be written as `'-.-..--...'`, which is the concatenation of `'-.-.'`, `'.-'`, and `'-...'`. We will call such a concatenation the transformation of a word.

Return the number of different transformations among all words we have.

1st Example

```
Input: words = ['gin','zen','gig','msg']
Output: 2
Explanation: The transformation of each word is:
             'gin' -> '--...-.'
             'zen' -> '--...-.'
             'gig' -> '--.--.-'
             'msg' -> '--.--.-'
             There are 2 different transformations:
             '--...-.' and '--.--.-'.
```

2nd Example

```
Input: words = ['a']
Output: 1
```

Constraints

- `1 <= words.length <= 100`
- `1 <= words[i].length <= 12`
- `words[i]` consists of lowercase English letters.

Solution

```
const uniqueMorseRepresentations = (words) => {
```

Solution continues on next page...

```
    const decode       = word => word
                                 .split('')
                                 .map(morseLetter =>
                                  morseLibrary[morseLetter])
                                 .join(''),
          morseLibrary = {
              a: '.-', b: '-...', c: '-.-.', d: '-..',
              e: '.', f: '..-.', g: '--.', h: '....',
              i: '..', j: '.---', k: '-.-', l: '.-..',
              m: '--', n: '-.', o: '---', p: '.--.',
              q: '--.-', r: '.-.', s: '...', t: '-',
              u: '..-', v: '...-', w: '.--', x: '-..-',
              y: '-.--', z: '--..'
          };

    return new Set(words.map(decode)).size;
};
```

Explanation

I've created a function called `uniqueMorseRepresentations` that takes an array of words as input. Its purpose is to return the number of unique morse code representations of those words.

Inside the function, there is a variable called `decode` which is assigned a function. This function takes a word as input and performs the following steps: it splits the word into an array of individual characters, maps each character to its corresponding morse code representation using the `morseLibrary` object, and then joins the morse code representations back into a single string.

Another variable called `morseLibrary` is declared and assigned an object that maps each letter of the alphabet to its corresponding morse code representation.

The function then creates a new `Set` object using the `new Set()` constructor. It does this by mapping the `decode` function to each word in the input array, which converts the words to their morse code representations. The `Set` object automatically eliminates any duplicate representations.

Finally, the function returns the `size` property of the `Set` object, which represents the number of unique morse code representations.

In summary, the `uniqueMorseRepresentations` function takes an array of words, converts each word into its morse code representation, eliminates duplicates, and returns the count of unique morse code representations.

Author: Trevor Morin