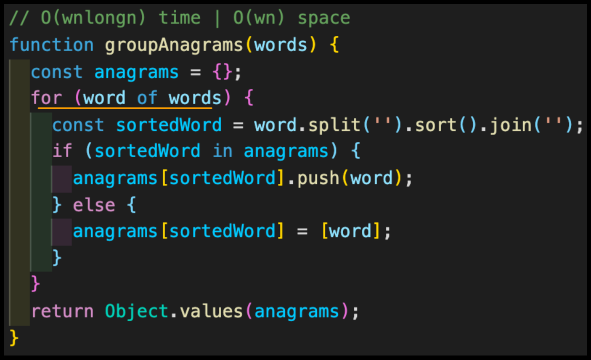
Input: Any array of strings:

Ex: words = ["yo", "act", "flop", "tac", "foo", "cat", "oy", "olfp"]

→ Anagrams are strings made up of exactly the same letters, where order doesn't matter

Output: A list of anagram groups in no particular order

Ex: [["yo", "oy"], ["flop", "olfp"], ["act", "tac", "cat"], ["foo"]]



```javascript
// O(wnlongn) time | O(wn) space
function groupAnagrams(words) {
  const anagrams = {};
  for (word of words) {
    const sortedWord = word.split('').sort().join('');
    if (sortedWord in anagrams) {
      anagrams[sortedWord].push(word);
    } else {
      anagrams[sortedWord] = [word];
    }
  }
  return Object.values(anagrams);
}
```

Idea: Use a hash table. At each word, we sort the word alphabetically. Then we check if the sorted word is alredy in our hash table. If it is, we add the current word to the sortedWord key in the hash table. Otherwise, we create a new key and add the word as an array

Time: O(w n log n) (where w is the # of words and n is the length of the longest word) since at each word (w), we sort it (nlogn) alphabetically

Space: O(wn) since we are returning an array of length wn

Note: for... of is used to iterate over the values in an iterable (like an array). The for... in is used to iterate over the properties of an object (the object keys). It can also be used to iterate over the index values of an iterable