Palindrome Permutation: Given a string, write a function to check if it is a permutation of a palindrome.

Ex) input: Tact Coa
output: true
("taco cat", "atco cta", etc).

A palindrome is a string that:
1) has at least two characters ("a" isn't a palindrome)
2) if it has a length that is an even number, then each recorded character must exist in the string an even number of instances
3) if it has an odd length, then one character is allowed to have an odd number of instances

ex) r a c e c a r } length = 7

r - 2
a - 2
c - 2
e - 1

this matches 3's criteria, and also tells us to use a dictionary, where the keys are the characters of the string, & the values are the number of instances of the key in the string.

Implementation:

```
isPermutationOfPalindrome(str) {
    if (str.length < 2) {
        return false;
    }

    let charMap = new Map();

    for(let i=0; i<str.length; i++){
        if(charMap.has(str[i])){
            charMap.set(str[i], charMap.get(str[i]) + 1);
        } else {
            charMap.set(str[i], 1);
```

```
    }
  }

  let nonEvenCount = 0;
  for(let value of charMap.values()){
      if(value % 2 !== 0){
          nonEvenCount++;
      }
  }
  if(str.length % 2 === 0){
      if(nonEvenCount > 1){
          return false;
      }else{
          return true;
      }
  }else{
      if(nonEvenCount === 1){
          return true;
      }
  }
  return false;
}
```

Complexity:

Time: $2n = O(n)$

Space: $O(n)$, due to the hash map