*High-level strategy:*

We use a stack. We check if the character is included as an opening bracket (because in order to have a “balanced” bracketing symbol, it must start with an opening bracketer.

When we encounter it, we add it to the stack. And if we encounter a corresponding closing bracket, we pop an element off the stack. If all bracketers have been accounted for, the stack should be empty. If any are not balanced, then the brackets are not balanced.

const balancedParens = input => {

const stack = [];

const ref = {

'}': '{',

')': '(',

'}': '{'

}

*// iterate through the string*

*// check the values of the reference object and determine if the character includes that character.*

*// in this case, we push the character into our stack.*

*// otherwise, if the character is contained as a key (a closing bracket),*

*// we will check if the last elemnt in the stack is the the value stored with that key (the closing bracket).*

*// if so, we'll pop the last el off the stack.*

*// and if not, we'll return false.*

*// if the stack is empty, then we return whether or not the stack is empty. If it's empty, we've got balanced brackets. If there's esomething left, we'll need to return false.*

}