JavaScript Coin Change

Challenge

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return `-1`.

You may assume that you have an infinite number of each kind of coin.

1st Example

```
Input: coins = [1,2,5], amount = 11
Output: 3
Explanation: 11 = 5 + 5 + 1
```

2nd Example

```
Input: coins = [2], amount = 3
Output: -1
```

3rd Example

```
Input: coins = [1], amount = 0
Output: 0
```

Constraints

- `1 <= coins.length <= 12`
- `1 <= coins[i] <= 2³¹ - 1`
- `0 <= amount <= 10⁴`

Solution

```
const coinChange = (coins, amount) => {
    const dp = Array(amount + 1).fill(amount + 1);

    dp[0] = 0;

    for (let coin of coins) {
        for (let i = coin; i <= amount; i++) {
            dp[i] = Math.min(dp[i], dp[i - coin] + 1);
        }
    }

    const ans = dp[dp.length - 1];

    return ans == amount + 1 ? -1 : ans;
};
```

Explanation

I've created a function called `coinChange` that takes in two parameters: an array of coins and an amount. The purpose of the function is to calculate the minimum number of coins needed to make up the given amount using the provided coins.

The function starts by initializing an array called `dp` with a length of `amount + 1` and fills it with the value of `amount + 1`. This array will be used to store the minimum number of coins needed for each amount.

Next, the value of `dp[0]` is set to `0`, indicating that no coins are needed to make up an amount of `0`.

The function then iterates through each coin in the `coins` array. For each coin, it further iterates through each amount starting from the value of the coin up to the given `amount`.

Inside the nested loops, the function updates the value of `dp[i]` by taking the minimum of its current value and the value of `dp[i - coin] + 1`. This calculation represents the minimum number of coins needed to make up the current amount `i`.

After the nested loops, the value of `dp[dp.length - 1]` represents the minimum number of coins needed to make up the given amount.

Finally, the function checks if the value of `ans` (which is equal to `dp[dp.length - 1]`) is equal to `amount + 1`. If it is, the function returns `-1`, indicating that it is not possible to make up the given amount with the provided coins. Otherwise, it returns the value of `ans`.

In summary, the `coinChange` function utilizes dynamic programming to calculate the minimum number of coins needed to make up a given amount using the provided coins. It returns `-1` if it is not possible to make up the amount with the given coins.

Author: Trevor Morin