Problem Statement:

In England the currency is made up of pound, £, and pence, p, and there are eight coins in general circulation:

1p, 2p, 5p, 10p, 20p, 50p, £1 (100p) and £2 (200p).

It is possible to make £2 in the following way:

1×£1 + 1×50p + 2×20p + 1×5p + 1×2p + 3×1p

How many different ways can £2 be made using any number of coins?

Solution:

Based on the possible ways mentioned in the problem, I created a sequence of combinations of currencies available in general circulation.

=> For 1P

combination
=>1 * 1p
combination
=> 2 * 1p
combination
=> 3 * 1p

…and that goes for other value pence(5p, 20p…)

=> For 2P

combination
=> 1 * 2p
combination
=> 2 * 1p
combination
=> (2 * 1p) + 1p

At the end (as we go higher with the currencies-> 5p, 10p) I realize ultimately we would end up in the same combination once we start breaking down these coins into 1penny , 2 pennies etc.

For example:
=> For 5P

combination
=> 5p
combination
=> 2×2p + 2x2p
combination
=> 5 x1p
combination
=> 1×2p + 1×2p + 1p
combination
=> 2×1p + 1p

With this I thought of iterating over the general circulation currency available that would help determine whether there is a combination possible.
I started with storing all currency in array :

```
var currency= [1, 2, 5, 10, 20, 50, 100, 200];
   var combinations = 0;

   for (var i of currency) {
        // iterate over the currency
   }
```

Based on the combination list, I thought of iterating over every currency one by one starting with smallest(i.e 1) and then incrementing it as we go.
(I recollect Depth first search uses similar approach- where we traverse one side of the tree- go deep through it until we reach the last element and then come back to the top.)

I tried using similar approach here- where we iterate through one set of currencies until we reach to the end of the currencies combination.

however I had to look up to see the DFS wiki page to see possible params it takes for iteration.
I came up with this:
Reference:
https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

```
if (start -  i >= 0 && i <= end) {
        combinations = combinations + getCombinations      (start – i, i);  // recursive function
```

```
}
```

Final solution:

```
function getCombinations(start, end) {
    if (start === 0){
 return 1; // only 1 combination present – i.e with itself
}

    var currency = [1, 2, 5, 10, 20, 50, 100, 200];
    let combinations = 0;
    for (var i of currency) { // iterate over the coins
        if (start- I >= 0 && i <= end) {


            combinations = combinations + getCombinations (start –
i, i);  // recursive
        }
    }
    return combinations;
}
```

Sources:
https://en.wikipedia.org/wiki/Depth-first_search
https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

HackerRank discussion board to check if I was thinking correctly or completely in wrong direction

Sample Out:

https://jsfiddle.net/L2yxh1m3/2/

Time taken:
~8-10 hours