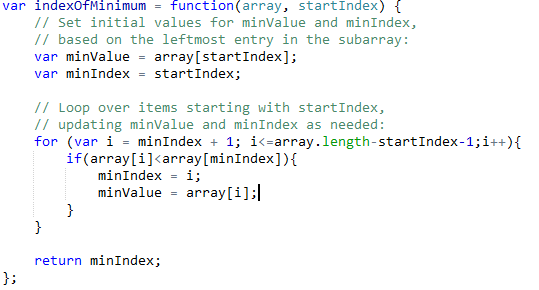
A key component of sorting is swapping the location of two components.

Here is how the selection sort algorithm works:

1. Find the smallest card. Swap it with the first card.
2. Find the second-smallest card. Swap it with the second card.
3. Find the third-smallest card. Swap it with the third card.
4. Repeat finding the next-smallest card, and swapping it into the correct position until the array is sorted.

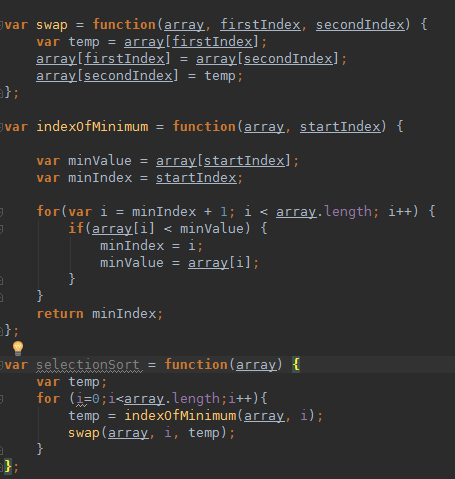
We basically look at our first card and do an array swipe to the end to find whether there is a smaller card than that. Once this is done, we do the same for the second card. The problem is that our algorithm has no memory, so we have to do the process of swiping all over again everytime (so it is O(∑n)=O(n²) ) .

Here is a Javascript function that takes an array of numbers, a minimum index, and loops through the subarray right of the minIndex to see if there is a index that array[index] <array[minIndex]



This is going to be the first step of the loop in our select sort algorithm. Once we found the index of the value that is smaller than the value we are currently analyzing, we are gonna swap them.

Here is how the whole algorithm looks like:



\*\*Here’s the general trick to sum up any sequence of consecutive integers:

1. Add the smallest and the largest number.
2. Multiply by the number of pairs.

(if uneven number of integers, pick out the middle one and count it as half a pair).

In other words, if there are ∑n steps to be done, it equals http://i.gyazo.com/4eaf3a66b557e197f1b09a7d4ecbffa2.png

So the complexity here is n² because we take only the biggest coefficient in the polynom.