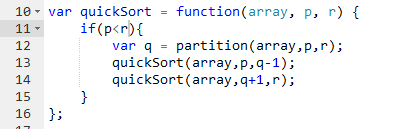
In merge sort, the divide hardly does anything, it’s with the merge that with sort the subarrays. With quick sort, it’s the opposite: the divide does all the work. The way it works is that in an array(p…r) we chose a pivot q and throw in to the left all the elements of array(p…r) that are smaller than the pivot q and throw in to the right of q all the elements of array(p…r) that are bigger than q. We usually take the right most element of an element to serve as our pivot q. Then we repeat this procedure (we call it partitioning) for both subarrays left and right of q, etc. until we arrive to the base case. There is no merge procedure then? no. Because our pivot can be the smallest or the biggest element already in the array(p…r), there is a possibility that one of our subarrays will be null. That means that the base case in quick sort is n=0.

This is the conquer part



And here is the partitioning part. There are three four groups: the values left of the pivot, which we know are smaller, the values right, which we know are bigger, the pivot, and the elements that have not yet been compared to the pivot. First we put the pivot on the left outmost index, and we increment our j index while comparing it to our outmost right index. If the current index has a smaller value than our pivot, we swap it with q (which is on the left side of the array) and increment q, so that the next smaller value than r goes left, but on the right of the current value). If array[j] is bigger, we leave it where it is. At the end, q has a certain value, which is the index of the last value moved to the left (because it was smaller than index [r]) +1. Everything is now in order.

