From Khan Academy:

<http://khan4019.github.io/front-end-Interview-Questions/sort.html#quickSort>

## Quick sort

**how does it works:**

**Step-1:** You have to pick a pivot. This could be randomly selected or the middle one. Here we select the last element of the array.

**Step-2:** Put all the items smaller than the pivot value to the left and larger than the pivot value to the right.

**Step-3:**Repeat the step-2 for both left and right side of the pivot (pick a pivot, put all item smaller than the pivot to the left and larger on the right)

#### Explain the code

**Call Quick sort:** Pass the array and pass left and right to the quickSort function. For the first call, left would be the index of the first element which is 0 and right would be the index of the last element which would be length -1.

**Select Pivot:** We select pivot as the last index of the array.

**Call Partition function:** After calculating the pivot, we send the pivot to the partition function. In the partition function we pass array, pivot index, left and right.

**partitionIndex:** In the partition function, we keep move all the items smaller than the pivot value to the left and larger than pivot value to the right. We have to keep track of the position of the partition. so that we can split the array into two parts in the next step. This tracking of the index where we partition the array is done by using partitionIndex variable. the initial value is left.

**Swap function:** This is just a helper function to swap values of the array.

**move elements:** we start a for loop from the left, and if the values is smaller than the pivot values we swap it with the position of the partitionIndex and increase the value of the partitionIndex. If the value is bigger, we don't do anything. We keep going until the element before the last element (remember last element is the pivot)

**place pivot** After moving all the smallest element to the left, we swap the last element (pivot value) with the partitionIndex. By doing this, the pivot sits where it suppose to sit when the full array is sorted. As all elements left to it smaller and all element right to it is bigger. End of the function partition, return the partitionIndex

**Repeat the process:**Now come back to quickSort function. when you get the partitionIndex, apply quickSort for the left side of the array and right side of the array. keep doing it until left is smaller than right.

function quickSort(arr, left, right){

var len = arr.length,

pivot,

partitionIndex;

if(left < right){

pivot = right;

partitionIndex = partition(arr, pivot, left, right);

//sort left and right

quickSort(arr, left, partitionIndex - 1);

quickSort(arr, partitionIndex + 1, right);

}

return arr;

}

function partition(arr, pivot, left, right){

var pivotValue = arr[pivot],

partitionIndex = left;

for(var i = left; i < right; i++){

if(arr[i] < pivotValue){

swap(arr, i, partitionIndex);

partitionIndex++;

}

}

swap(arr, right, partitionIndex);

return partitionIndex;

}

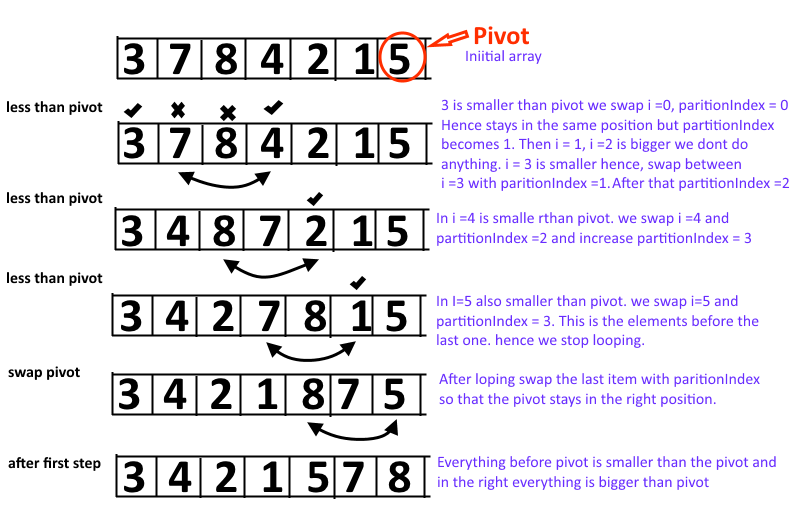
function swap(arr, i, j){

var temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}



quickSort([11,8,14,3,6,2,7],0,6);

//[2, 3, 6, 7, 8, 11, 14]

quickSort([11,8,14,3,6,2,1, 7],0,7);

//[1, 2, 3, 6, 7, 8, 11, 14]

quickSort([16,11,9,7,6,5,3, 2],0,7);

//[2, 3, 5, 6, 7, 9, 11, 16]

ref: [quick sort](https://gist.github.com/paullewis/1981455)

**time complexity:** talk about time complexity of quick sort