JavaScript Algorithms and Data Structures Masterclass

# Section 10-13: Intro to Elementary Sorting: Bubble Sort, Selection Sort, Insertion Sort

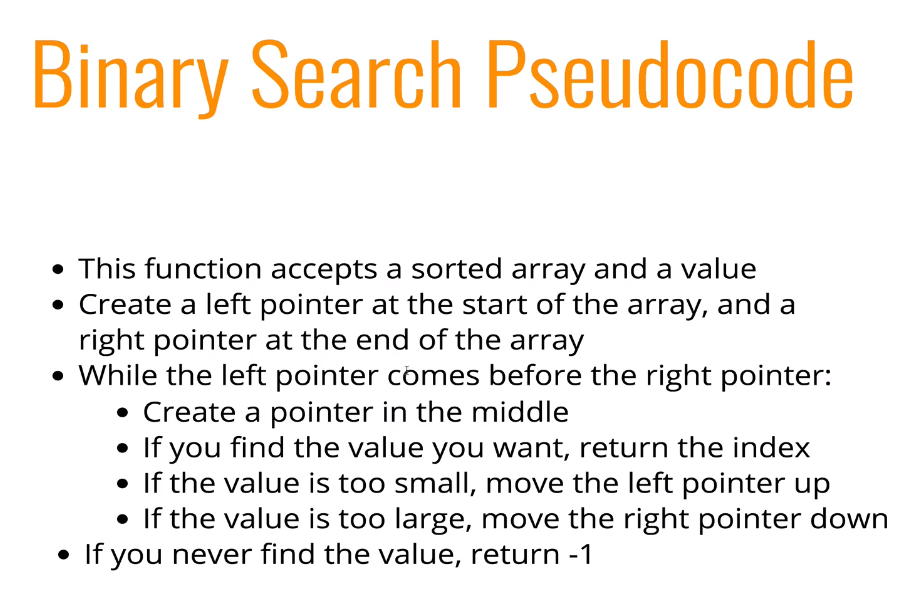
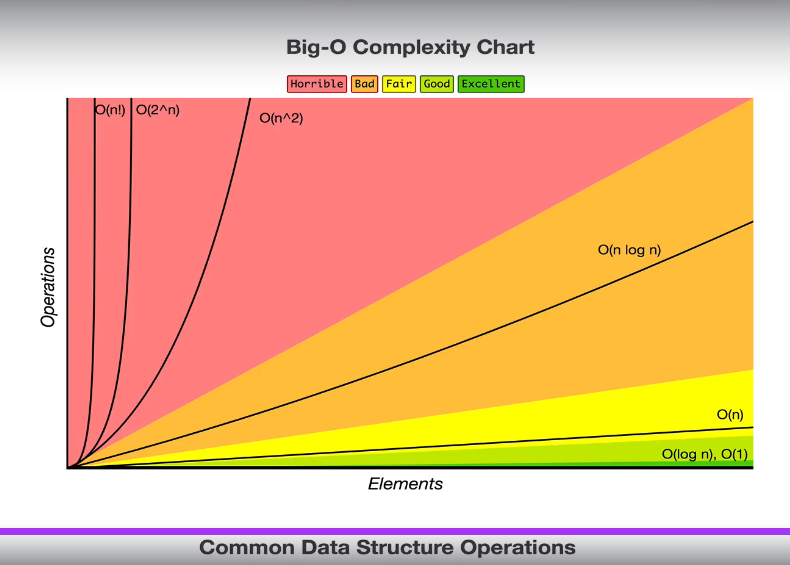
## Objectives

* Implement Linear Search on arrays
* Implement Binary Search on sorted arrays
* Implement naive string search
* Implement KMP string search

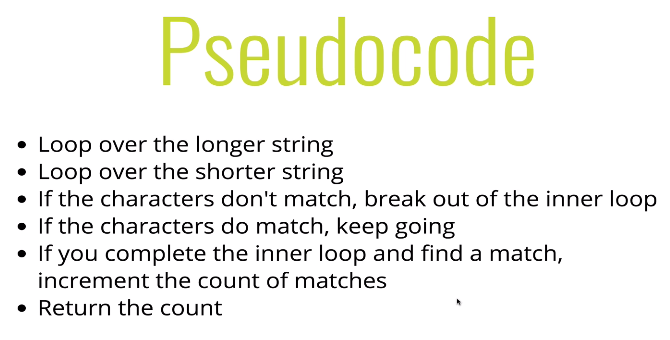
## Linear Search

* Can be useful in ‘unsorted’ arrays
  + Go through each individual element in an array to check linearly
    - (**ex**. .indexOf, .includes, .find, .findIndex)
* Time Complexity = O(n)

## Binary Search

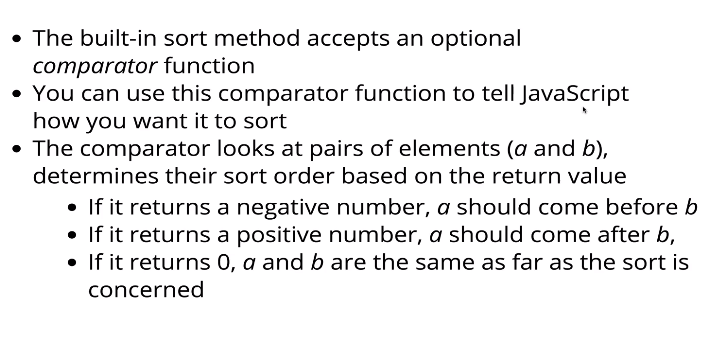
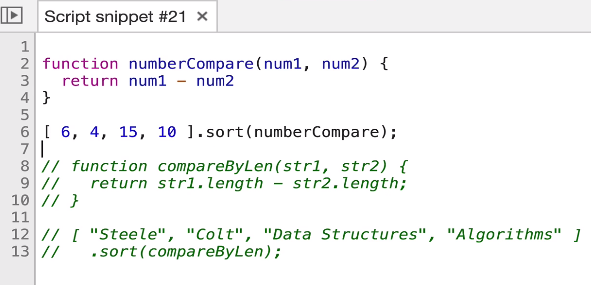
* Only works on sorted arrays
* Eliminate half of the remaining elements at a time (faster than Linear Search)
  + aka: Divide and Conquer
  + 
* **Big O of Binary Search**
  + O(log n) – worst/avg. Case
    - Every time the array doubles, 1 extra step is added
  + Best case: O(1)
  + 

## Naive String Search

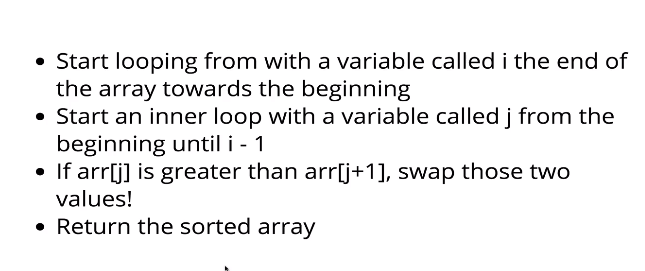
* This looks for a sequence of characters in a longer string
  + **ex**. find “omg” in “wowomg”
    - 

# Section 11: Bubble Sort

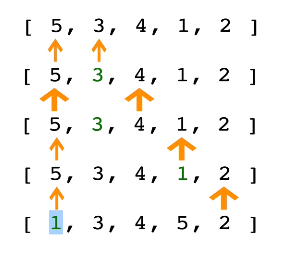
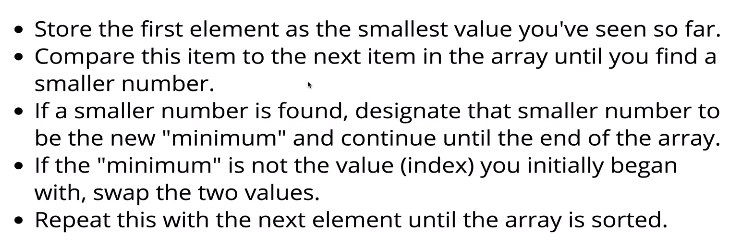
## Elementary Sorting Algos:

* Sorting is...
  + Rearranging the items in a collection (ex. Array) so they have an order
    - **ex**. Sort nums by size
* Built-In JS .sort method won’t sort properly (by unicode)
  + You must give it a comparator method for it to sort properly:
    - 
    - 

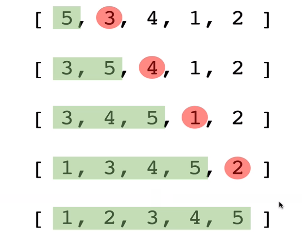
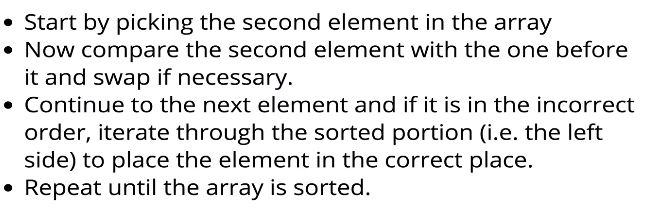
## Bubble Sort Implementation and Optimization

* Bubble the largest values to the top
  + Check the adjacent value, swap if the value is bigger than the other
  + Keep swapping till all the largest values are ordered:
    - 

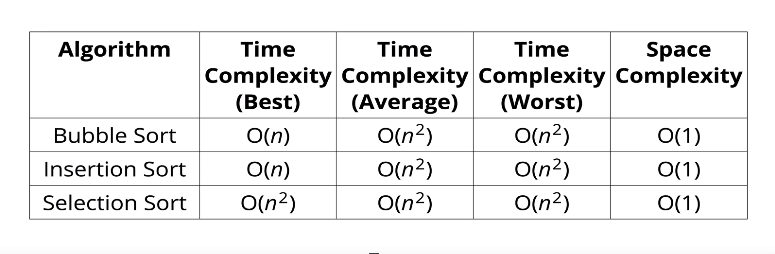
# Section 12: Selection Sort

* Similar to Bubble Sort, but instead:
  + Place smallest values into sorted position first, instead of largest
    - 
      * Place the smallest value in Index0/ the starting point of the loop
* **Psuedo-code**:
  + 

# Section 13 – Insertion Sort

* Builds up the sort by gradually creating a larger left half which is sorted:
  + 
  + **Pseudo-code:**
    - 
* **Time Complexity**
  + Worst Case: O(n2)
  + Best Case: O(n)

## Comparison of Elementary Sorting Algorithms:

* 
  + AKA: Quadratic Sorting Algorithms