Week 2 - Arrays 1 - Wednesday Recap

Binary Search

Given sorted array, return whether a value is present. Do not sequentially iterate: 'divide and conquer', taking advantage of the fact that the array is sorted.

```
function binarySearchArr(arr, val)
{
  var startIdx = 0;
  var endIdx = arr.length;
  while (endIdx > startIdx) {
    var midIdx = Math.floor(
                 (startIdx + endIdx) / 2);
    if (val > arr[midIdx]) {
      startIdx = midIdx + 1;
    } else if (val < arr[midIdx]) {
      endIdx = midIdx;
    } else { return true; }
  }
  return false;
}
```

Min Of Sorted-Rotated

Find the min value in a sorted-then-rotated array.

```
function minIdxOfSortedRotatedArr(arr)
{
  var startIdx = 0;
  var endIdx = arr.length;
  while (endIdx - startIdx > 1) {
    var midIdx = Math.ceil(
                 (startIdx + endIdx) / 2);
    if (arr[startIdx] < arr[midIdx]) {
      startIdx = midIdx;
    } else { endIdx = midIdx; }
  }
  return startIdx;
}
```

Rotate Array

Implement rotateArr(arr, shiftBy) that moves arr values to the right by *shiftBy* spaces. Values that are shifted off the array's end should 'wrap-around' to appear at the array's beginning, so no data is lost.

Optionally, add these advanced features:
a) negative shifts (shift left instead of right),
b) minimize how much you touch each element,
c) minimize memory usage; a few local vars are OK, but create a solution that handles arrays as long as a million, with shiftBy of similar magnitude.

```
//  Handles negative and very large shiftBy
//  values, but is an inefficient O(n2)
//  solution. More about O(n2) later....

function rotateArr(arr, shiftBy)
{
  if (!arr.length || !shiftBy) { return; }

  while (shiftBy < 0) {
    shiftBy += arr.length;
  }
  shiftBy %= arr.length;

  while (shiftBy--) {
    var temp = arr[arr.length - 1];
    for (var idx = arr.length - 2;
         idx >= 0; idx--) {
      arr[idx+1] = arr[idx];
    }
    arr[0] = temp;
  }
}
```

Week 2 - Arrays 1 - Thursday

This week you will familiarize yourself with basic array manipulation. Here is a list of methods to study; some or all will be used in this week's challenges.

for / while loops	*array.pop() & push()*	*can contain different data types*
if / else statements	*arrays grow: arr.length == lastIdx-1*	*arrs are objects, passed by reference (ptr)*

Second-to-Last

Given an array, return the second-to-last element.

Answer:

Nth-to-Last

Return the element that is N-from-array's-end.

Answer:

Second-Largest

Given an array, return the second-largest element.

Answer:

Nth-Largest

Given an array, return the Nth-largest element: there should be (N - 1) array elements with larger values.

Answer:

Tomorrow: the time-space continuum