JS 12

In JS 11, we took data from a search bar, stored it into an array, and then loop from the start of the array at index 0 until the end of the array to find the average, highest, and lowest values.

While inputing the numbers, we noticed that **the numbers are out of order**. There are situations where we want the numbers to be in order from lowest number to highest number or the opposite.

For example, what if we want to orgnize a directory (ie. a folder on our computer) according to "**File Size**". This allows us to see the files that are smallest in size first while the files that are largest in size are last.

Another example is organizing a directory according to "**Date Modified**". This allows us to see the most recent file first and the oldest files last.

The two examples above are important because, **when activated, the two features allow us to be more productive**. Instead of manually remembering when we saved a file, we simply just press "**Date Modified**" and the computer does the work for us.

In this session, we learn how to organize the data even more by using array manipulation to implement simple sorting.

1. array manipulation 2. simple sorting 3. push()

==========

Continue to the next page for instructions

0. **DO NOT WRITE THE CODE BELOW IN BETWEEN** `<script> </script>.` Go to the **GREEN** and make your code looks like the **GREEN CODE ONLY**

```
// -- jan. Using "m1"
function iDV()
{
   var uInput = document.getElementById( "nEInput" );

    var eData = uInput.value;

    alert( "user entered: " + eData );

    if( eData == "" )
    {
         alert( "Can't enter blank data. Try again" );
    }
    else
    {
         alert( "numArr: " + numArr );
         //numArr.push( eData );
         insert_and_sort( parseInt(eData) );

        var ePTag = document.getElementById( "ePTag" );

        // -- clear the tag's content
        ePTag.innerHTML = "";
        for( var i = 0; i < numArr.length; i++ )
         {
                ePTag.innerHTML += numArr[i] + " ";
         }



    }

}
```

1. Write JS Code In Between <script> </script>. ONLY write the code in **RED** and position it under the code in **BLACK**

```
var numArr = [];
var lowest = 0;
var highest = 0;

// -- sort
var tempArr = [];
var hit     = 0;

function insert_and_sort( newData )
{
	// -- if this is the start, then go here
	if( numArr.length == 0 )
	 {
		alert( "first item" );
		numArr.push( newData );
	 }

	 // -- newData is put at the beginning
	 else if ( newData < numArr[0] || newData == numArr[0] )
	 {
		alert( "new beginning" );

		// -- clear the array
		tempArr = [];

		// -- make new beginning
		tempArr[0] = newData;

		// -- push data at the end
		move_a_to_b( 0, numArr.length );



		numArr = tempArr;
		alert( "numArr: " + numArr );
	 }

	 // -- new Data is put at the end
	else if( newData > numArr[numArr.length-1] || newData == numArr[numArr.length-1] )
	 {
		alert( "new end" );
		numArr.push( newData );
	 }

	 else
	 {
		// -- find the pivot point and then move data
		alert( "second and beyond" );
		for( var i = 0; i < numArr.length && !hit; i++ )
		{
		if( newData < numArr[i] || newData == numArr[i] )
		{
			hit = 1;
```

```
                tempArr = [];

                move_a_to_b( 0, i );
                tempArr[i] = newData;
                move_a_to_b( i, numArr.length );
        }
        }


        numArr = tempArr;
        hit = 0;


    }



}


function move_a_to_b( start, end )
{
       alert( "inside move_a_to_b()" );
        alert( "end: " + end );
       for( var i = start; i < end; i++ )
        {
              tempArr.push( numArr[i] );

        }


}
```

Explanation

There are 4 cases that we will talk about
1. what happens if the array is empty
2. what happens if the number that we entered is the smallest number
3. what happens if the number that we entered is the largest number
4. what happens if the number that we entered has to be put in the middle of the array

Case 3 is simple and will not be talked about

Case 1 is the image with the red 1 and square box.

Case 2 & 4 are combined into a single explanation. It is the red 4 with red square box.

```
// -- if this is the start, then go here
if( numArr.length == 0 )
{
        alert( "first item" );
        numArr.push( newData );
}
```

since the array has a length of 0, it is empty.
1. the front and back are the same

1

An array is a line.
1. After we are done, we exit at the front of the line
2. If we want to enter the line, we enter at the back

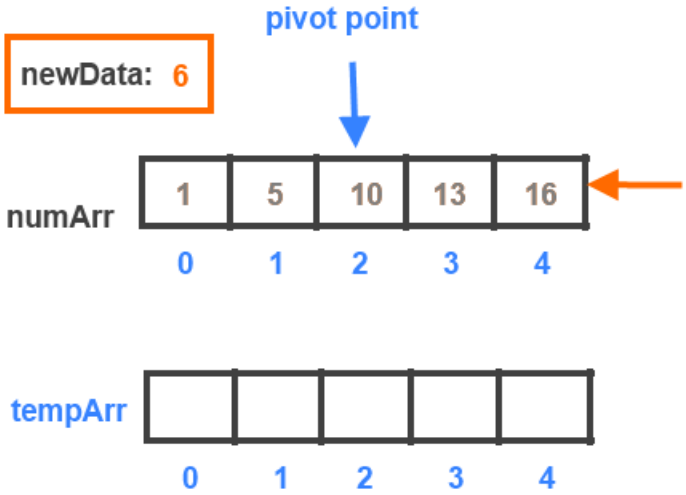
The **.push(newData)** is a special operation to enter data at the back of the line

4

The next step is to find the **pivot point**.

In our case, the **pivot point** is the point where we will put in the new data.

Once we find the pivot point, we just loop and push at the end of a new array to save the data and make space for new data

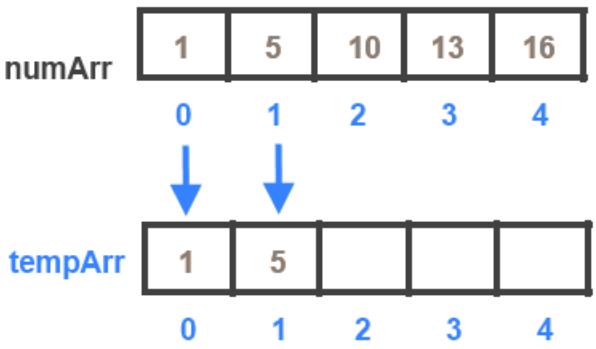


Steps
1. start at index 0 and check if the data inside index 0 is less than the value stored inside newData

2. increment to the next index until we find a spot where newData is less than the value stored inside the array

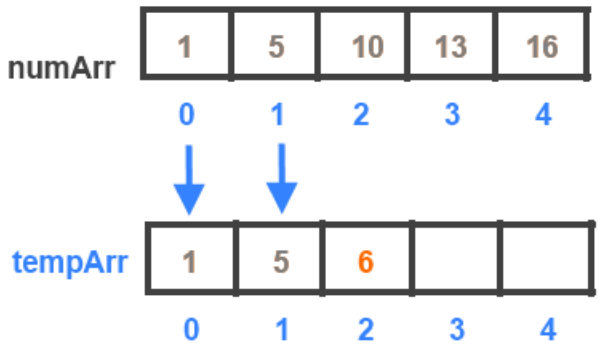
3. position 2 satisfies our check condition

3.1. position 2 is our pivot point



4. everything to the left of the **pivot point** we want to save into a new array called tempArr

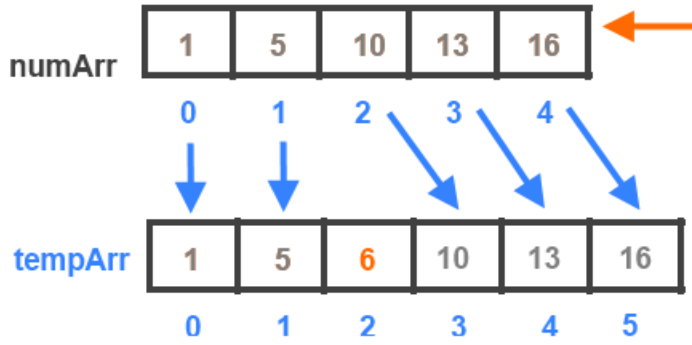
4.1. we use a "for" loop and square brackets to make the move



5. put the newData, which is 6, into the position of the pivot point of the tempArr

5.1. if we put newData into the pivot point of numArr, we lose the existing data

5.2. SHAME!!!



6. we finish by moving everything to the right of the pivot point into the new array

6.1. we can use **push** because we are putting it at the end

What is the Modulus Operator (%)

Modulus operator is the percent sign (%). This operator is important because it gives us the remainder of a division between two numbers. The modulus is used to determine if a number is even or odd.

What is an even number? An even number is a number that is divided by the number 2 and has a **remainder of 0**. In essence, an even number cuts itself in half and there is no remainder left.

What is an odd number? An odd number is a number that is divided by the number 2 and has a **remainder of 1**. In essence, an odd number cuts itself in half and **DOES** have at least one part as a spare.

```
var remainder = 4 % 2;      // -- modulus operator gives us a remainder of 0

var remainder = 5 % 2;      /* -- 5 is an odd number

                                  modulus operator gives us a remainder of 1.
                                  So, the number 5 is an odd number

                            */
```

HTML Challenge. Inside <body> </body>
1. create another button called "**Odd Nums Only**"
2. when we click on this button, a JS function called "**show_odd()**" is called
2.1. use "**onclick**" to have HTML send a signal to JS code

JS Challenge. Inside <script> </script>
1. Write the JS function definition called "**show_odd()**".
2. Inside the function definition, write a "**for**" loop that only prints out the odd numbers found within the array named `numArr`
    2.1. you will need to use the modulus operator and **check if the remainder is a 1**
    2.2. do we need to use the array named `tempArr` to hold the numbers that are odd?
3. outside of the "**for**" loop, add this code below

```
numArr = tempArr;
var ePTag = document.getElementById( "ePTag" );

// -- clear the tag's content
ePTag.innerHTML = "";
for( var i = 0; i < numArr.length; i++ )
{
      ePTag.innerHTML += numArr[i] + " ";
}
```

HTML Super Challenge. Inside <body> </body>

1. create another button called "**Largest First**"
2. when we click this button, a JS function called "**show_LF()**" is called

2.1. use "**onclick**" to have HTML send a signal to JS code

JS Super Challenge. Inside <script> </script> 20NB.

1.Think about the array and how it is organized ... DO NOT ASK PARENTS FOR THE ANSWER!!!

1. Write the JS function definition called "**show_LF()**".
2. Inside the function definition, write a "**for**" loop that shows the largest number first, then the second largest number, then the third largest number, and ect.

    2.1. HINT: can we go in the reverse direction?

    2.2. you can use the array `tempArr` if you want but you don't have to …

3. outside of the "**for**" loop and ***if you use tempArr***, then add this code below

```
numArr = tempArr;
var ePTag = document.getElementById( "ePTag" );

// -- clear the tag's content
ePTag.innerHTML = "";
for( var i = 0; i < numArr.length; i++ )
{
       ePTag.innerHTML += numArr[i] + " ";
}
```