JS 11

An array is basically a line. Each person in line has a number and the first number is **ALWAYS** 0. When the number in your possession is called, then it is your turn and you can begin service.

There are many applications of an array. One simple example is a calculator. The numbers that we enter on the keyboard are put into a line and this is how a calculator uses the numbers in the line to calculate a result.

1. search bar 2. array manipulation 3. simple calculator

==========

1. Write HTML code in between <body> </body>. ONLY write the code in SKY BLUE and PURPLE. Next, position the code in **SKY BLUE and PURPLE AFTER** the code in **black**

```
<p id = "ePTag">


</p>


<div
class = "mathBlock"
>
      <button
      id      = "avgB"
      onclick = "calc_avg()"
      >
      Avg
      </button>

      <p
      id = "average"
      class = "result"
      >
      average
      </p>
</div>


<div
class = "mathBlock"
>
      <button
      id      = "highestB"
      onclick = "find_highest()"
      >
      Highest
      </button>
```

```
        <p
        id = "highest"
        class = "result"
        >
        hi
        </p>

</div>


<div
class = "mathBlock"
>
        <button
        id      = "lowestB"
        onclick = "find_lowest()"
        >
        Find Lowest
        </button>

        <p
        id = "lowest"
        class = "result"
        >
        low
        </p>

</div>
```

Explanation

Remember that the `<div> </div>` is the "**division**" tag that is used to "**divide the website into sections**". In the above code, we have 3 `<div>` tags and that means that there are at least 3 sections.

In each section, we have a button that relates to a math operation and then a place holder for the result.

Notice the code in purple. This is our event and action (ie. event handler). **The event** is "**onclick**" and the **action is a JS function**. You will write the JS function in the next chapter. This is important because it allows HTML to send a signal to JS code to execute an action.

For example, we have **the event** "`onclick`" attached to the word "**Avg**". When we click on "**Avg**", HTML sends a signal to JS code. In this case, the signal is "`calc_avg()`" and it is a JS function definition.

The HTML "`onclick`" is the event and the JS code is the action. We will write the JS code next.

2. Write the JS Code in between `<script> </script>`. ONLY WRITE THE CODE IN RED and position it after the variable declaration (ie. put the **red code** under the **black code**).

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

function find_highest()
{


}


function find_lowest()
{
	var lowObj = document.getElementById( "lowest" );
	lowest = parseInt( numArr[0] );

	for( var i = 0; i < numArr.length; i++ )
	{
		if( lowest > parseInt( numArr[i] ) )
		{
		lowest = parseInt( numArr[i] );
		}
	}

	lowObj.innerHTML = lowest;

}

function calc_avg()
{
	var sum = 0;

	for( var i = 0; i < numArr.length; i++ )
	{
		sum += parseInt( numArr[i] );
	}

	alert( "sum: " + sum );

	var avgObj = document.getElementById( "average" );

	alert( "len of numArr: " + numArr.length );
	avgObj.innerHTML = sum/numArr.length;


}
```

Explanation

The 3 function definitions match what we wrote in HTML. For example, the JS function name **"`calc_avg()`"** matches the HTML code. Look below and confirm that **clicking** on **"Avg"** will send a signal to JS code to call the action **"`calc_avg()`"**.

HTML Code

```
<div
class = "mathBlock"
>
      <button
      id      = "avgB"
      onclick = "calc_avg()"
      >
      Avg
      </button>

      <p
      id = "average"
      class = "result"
      >
      average
      </p>
</div>
```

JS Code

```
function calc_avg()
{
    var sum = 0;

    for( var i = 0; i < numArr.length; i++ )
    {
        sum += parseInt( numArr[i] );
    }

    alert( "sum: " + sum );

    var avgObj = document.getElementById( "average" )

    alert( "len of numArr: " + numArr.length );
    avgObj.innerHTML = sum/numArr.length;


}
```

*** You already wrote the code below. Transition to the Explanation paragraph below ***

```
function calc_avg()
{
      var sum = 0;

      for( var i = 0; i < numArr.length; i++ )
      {
            sum += parseInt( numArr[i] );
      }

      alert( "sum: " + sum );

      var avgObj = document.getElementById( "average" );

      alert( "len of numArr: " + numArr.length );
      avgObj.innerHTML = sum/numArr.length;


}
```

Explanation

The function definition `calc_avg()` loops through the array and adds the numbers together. Remember that we are using a search bar to enter the number. This is important because a search bar sees the input as letters (ie. characters) instead of numbers.

For example, if we type in 5 and then 3 into the search bar and then tell the computer to add the two numbers, **THE RESULT IS NOT 8**. Since a search bar sees letters, **the result is "53".** Why? Javascript uses the plus operator (+) as a math operation **OR** the concatenation operation. Concatenation means to join or combine together.

How does Javascript know which one to use, math operator or concatenation operator? The data will determine the operator to use.

So, if we say “5” + “5”, the **two operands are letters** and so Javascript uses the **concatenation operator.**

However, if we convert the letters into numbers using `parseInt()`, it would be the **number 5 + number 5** and Javascript uses the **math operator for addition**.

Looking at the code above,

```
sum += parseInt( numArr[i] );
```

Notice that we use the square bracket and then use a variable called `i` as a digital key to get the data at that index.

Next, we use **+=** to add the numbers together. We add numbers together one at a time until we get the total sum.

3. Write CSS code in between <style> </style>. Change the style of the blocks below to be any style you like.

```
body
{

	background-color: #006400;

}

p#ePTag
{
	margin		: 10px 0px 20px 0px;
	padding		: 0px 0px 0px 0px;
	color		: #ffd700;
	font-size	: 19px;

}


.mathBlock
{
	margin	: 0px 0px 20px 0px;
	border	: 1px solid #c0c0c0;

	text-align: center;
	background-color: #a0a0a0;
	background-image: url(https://vuongducnguyen.com/images/wLand.png);

}
```

JS Challenge 1.
1. write the JS function to keep track of the highest number. **The function definition is already there**. All that is needed is to write the code in between the curly braces

JS Challenge 2.
1. notice that if we enter invalid data, such as "**...**", " ", or "**hi**", the our simple calculator **STILL** accepts invalid data. This is not what we want.
2. we must filter the data. For simplicity, the accepted data are numbers between the range -50 and 50.
3. when the user clicks on the button "**Enter Data**", the function "**iDV()**" is called. ***** Inside the function, check if the data is a number between -50 and 50 *****

    3.0. use `parseInt()` to convert a representation into a number

    3.1. check if the data is between -50 and 50.

        if the check is true, then continue with the computation

        else, alert the user with the message **"only numbers between -50 and 50 are accepted"**