Day 3

Yogesh Yadav

Arrays

- **Arrays are collections and are ZERO indexed.** This means the first element is at index ZERO and the last element is at index arrayObject.length - 1. length property of the array object returns the size of the array.

- **The following JavaScript code creates an empty array.** length property returns 0 in this case.

  ```
  var emptyArray = [];
  alert(emptyArray.length);
  ```

  Output : 0

- **Another way to create an array is by busing the Array constructor as shown below.** In this example we are setting the length of the array to 10.

  ```
  var myArray = new Array(10);
  alert(myArray.length);
  ```

  Output : 10

- **Retrieving first and last elements from the array using the array index**
  ```
  var myArray = [10, 20, 30];
  ```
- document.write("First element = " + myArray[0] + "
");
- document.write("Last element = " + myArray[myArray.length - 1] + "
");
-

  Output :
  First element = 10
  Last element = 30

- for (var i = 0; i < evenNumbersArray.length; i++)
- {
- document.write(evenNumbersArray[i] + "
");
- }

- **Populating an array in JavaScript :** There are several ways to populate an array in JavaScript.

Declaring an array first and then populating using the array index

```
var myArray = [];

myArray[0] = 10;
myArray[1] = 20;
myArray[2] = 30;

alert(myArray);
```

Output : 10, 20, 30

Declaring and populating the array at the same time

```
var myArray = [10, 20, 30];
alert(myArray);
```

Output : 10, 20, 30

Array push() and pop()

we are populating **myArray** using a for loop and the array index.
Subsequently we are using another for loop to retrieve the elements from the array.
Finally we are displaying the length of the array using JavaScript alert.

```
var myArray = [];


for (var i = 0; i <= 5; i++)
{
    myArray[i] = i * 2;
}


for (var i = 0; i <= 5; i++)
{
    document.write(myArray[i] + "<br/>");
}

alert(myArray.length);
```

Please note : Retrieving array elements using the array index, will not change the length of the array.

- **JavaScript push() method**
  This method adds new items to the end of the array. This method also changes the length of the array.

  JavaScript pop() method
  This method removes the last element of an array, and returns that element. This method changes the length of an array.

- we are using push() method to populate the array and pop() method to retrieve elements from the array. Notice that push() and pop() methods change the length property of the array.

```
var myArray = [];


for (var i = 0; i <= 5; i++)
{
    myArray.push(i * 2);
}

alert(myArray.length);
```

```
for (var i = 0; i <= 5; i++)
{
    document.write(myArray.pop() + "<br/>");
}


alert(myArray.length);
```

JavaScript unshift() Method

push() method adds new items to the end of the array. To add new items to the beginning of an array, then use unshift() method. Just like push() method, unshift() method also changes the length of an array

Example :

```
var myArray = [2, 3];

// Adds element 4 after element 3
myArray.push(4);

// Adds element 1 before element 2
myArray.unshift(1);

document.write("Array elements = " + myArray + "<br/>");
document.write("Array Length = " + myArray.length);
```

Array elements = 1,2,3,4
Array Length = 4

- **JavaScript shift() Method**
  pop() method removes the last element of an array, and returns that element. shift() method removes the first item of an array, and returns that item. Just like pop() method, shift() method also changes the length of an array.

```
var myArray = [1, 2, 3, 4, 5];

// removes the last element i.e 5 from the array
var lastElement = myArray.pop();
document.write("Last element = " + lastElement + "<br/>");

// removes the first element i.e 1 from the array
var firstElement = myArray.shift();
document.write("First element = " + firstElement + "<br/><br/>");

document.write("Array elements = " + myArray + "<br/>");
document.write("Array Length = " + myArray.length);
```

Last element = 5
First element = 1

Array elements = 2,3,4
Array Length = 3

Array sorting

- Sorts the elements of an array. By default, the sort() method sorts the values by converting them to strings and then comparing those strings. This works well for strings but not for numbers.

```
var myArray = ["Sam","Mark","Tom","David"];
myArray.sort();
document.write(myArray);
```

Output : David,Mark,Sam,Tom

```
var myArray = [20, 1 , 10 , 2, 3];
myArray.sort();
document.write(myArray);
```

Output : 1,10,2,20,3

Notice that the numbers are not sorted as expected. We can fix this by providing a**"compare function"** as a parameter to the sort function. The compare function should return a negative, zero, or positive value.

Example :

```
var myArray = [20, 1, 10, 2, 3];
myArray.sort(function (a, b) { return a - b });
document.write(myArray);
```

how the compare function work. The function has 2 parameters (a,b). This function subtracts a from b and returns the result. If the return value is

Positive	a is a number bigger than b
Negative	a is a number smaller than b
ZERO	a is equal to b

So, based on these return values, the numbers in the array are sorted.

Sorting the numbers in descending order : There are 2 ways to sort an array in descending order

1. Return (b-a) from the compare function instead of (a-b)

Example :

```
var myArray = [20, 1, 10, 2, 3];
myArray.sort(function (a, b) { return b - a });
document.write(myArray);
```

Output : 20,10,3,2,1

2. Sort the numbers first in ascending order and then use the reverse function to reverse the order of the elements in the array.

Example :

```
var myArray = [20, 1, 10, 2, 3];
myArray.sort(function (a, b) { return a - b }).reverse();
document.write(myArray);
```

Output : 20,10,3,2,1

- **JavaScript reverse method :** reverses the order of the elements in an array.

  JavaScript splice method : This method is used to add or remove elements from an array.

  Syntax : array.splice(index,deleteCount,item1,.....,itemX)

index	Required. Specifies at what position to add or remove items
deleteCount	Required. The number of items to be removed. If set to 0, no items will be removed.
item1,.....,itemX	Optional. The new item(s) to be added to the array

Example :

```
var myArray = [1,2,5];
myArray.splice(2, 0, 3, 4);
document.write(myArray);
```

Output : 1,2,3,4,5

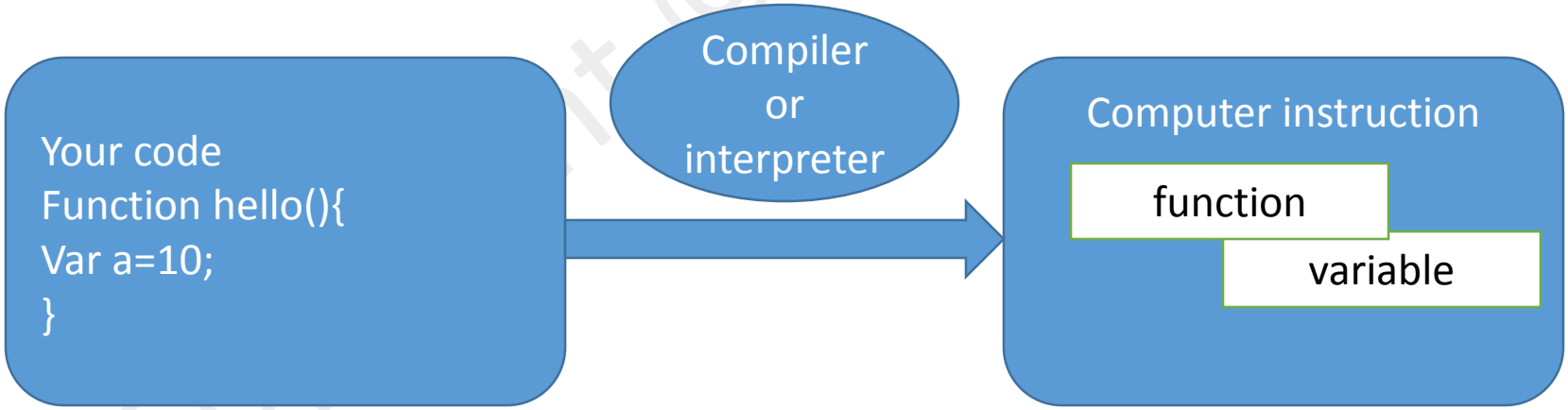
```
var myArray = [1,2,55,67,3];
myArray.splice(2, 2);
document.write(myArray);
```

Output : 1,2,3

Syntax Parser

- Syntax Parser: A program that reads your code and determines what it does and if its grammar is valid

- Your code isn't magic. Someone else wrote a program to translate it for the computer



Lexical Environment

- Lexical Environment: Where something sits physically in the code you write.
- 'Lexical' means 'having to do with words or grammar'. A lexical environment exists in programming languages in which **where** you write something is **important**.
- Where function variable sit in computer memory and how they are interacting with other variable and function that thing also handle by compiler or interpreter.

Execution context

- Execution context: A wrapper to help manage the code that is running
- There are lots of lexical environments. Which one is currently running is managed via execution contexts. It can contain things beyond what you have written in your code.