Pakistan Blockchain Institute

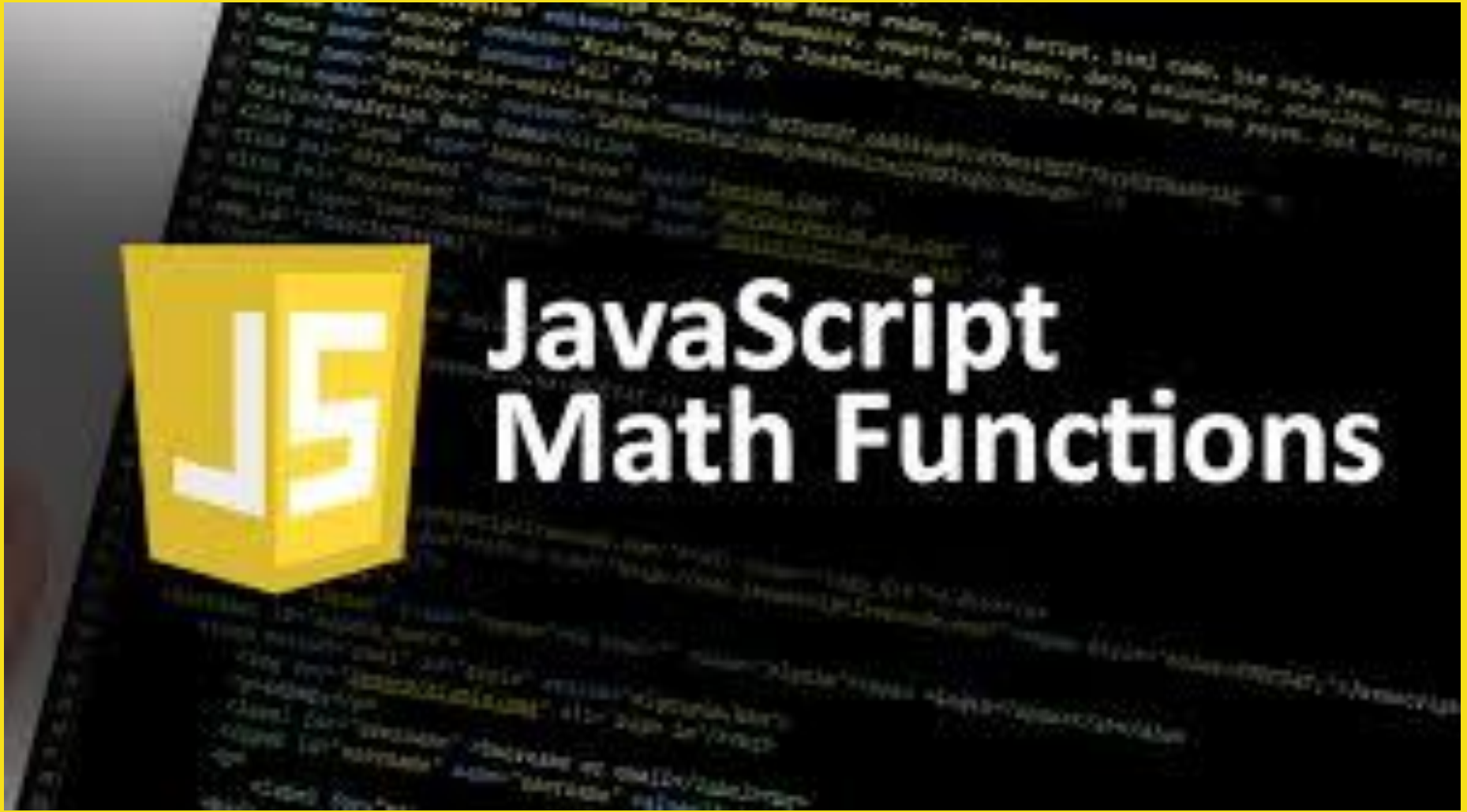
MODULE-1

JAVASCRIPT CRASH COURSE

Class-6

Edversity.

Raja Rizwan Saleem
Lead Blockchain Trainer


JavaScript
Math Functions

Math

Math class provides many functions that allows you to perform mathematical tasks on numbers

Math.round() function

Math.round() function

1. Math.round(x) returns the value of x rounded to its nearest integer
2. E.g calculating average score will result number with decimal places and you need to round them

```
var average = (15 + 23 + 39) /3 ; // 25.6666
var roundedAverage = Math.round(average); //
26  console.log(roundedAverage);
```

Math.round() function

```
var a = Math.round(4.7);     // 5
var b = Math.round(4.1);     // 4
var c = Math.round(4.5);     // 5
var d = Math.round(-4.1);    // -4
var e = Math.round(-4.7);    // -5
var f = Math.round(-4.5);    // -4
var g = Math.round(5);       // 5
```

Math.ceil() function

1. Math.ceil(x) returns the value of x rounded **up** to its nearest integer

```
var a = Math.ceil(4.7);     // 5
var b = Math.ceil(4.1);     // 5
var c = Math.ceil(-4.1);    // -4
var d = Math.ceil(-4.7);    // -4
```

Difference between Math.round() & Math.ceil()

Math.ceil() and **Math.round()** methods differ in a way that the former round off a number to its nearest integer in upward direction of rounding(towards the greater value) whereas the latter round off a number to its nearest integer in downward direction of rounding(towards lower value)

```
var a = 9.4889;
var b = Math.round(a);
console.log(b);// will Return 9
```

```
var a = 9.4889;
var b = Math.ceil(a);
console.log(b);// will Return 10
```

Math.floor() function

1. Math.floor(x) returns the value of x rounded **down** to its nearest integer

```
var a = Math.floor(4.7);     // 4
var b = Math.floor(4.1);     // 4
var c = Math.floor(-4.1);    // -5
var d = Math.floor(-4.7);    // -5
```

Math.random() function

1. Suppose you want to simulate the throw of a die. In the simulation, you want it to randomly come up 1, 2, 3, 4, 5, or 6
2. If you want build a game that will allow user to guess a number
3. Math.random() returns a random number between 0 (inclusive), and 1 (exclusive)
4. Everytime you execute this function it will return random value

Math.random() function

```
var num = Math.random();
// result will be like 0.5251908043871081
```

If you want to generate random number between some range then you have to add some calculations like:

```
var num = Math.random();
var num2 = (num * 6) + 1;

var dice = Math.floor(num2); // 1 to 6
```

Other Math functions

There are few more string functions you can learn

1. Math.pow()
2. Math.sqrt()
3. Math.abs()
4. Math.sin()
5. Math.cos()
6. Math.min()
7. Math.max()
8. Math.exp()
9. Math.log()

Controlling the length of decimals

1. In arithmetic operation you may face numbers with many decimal place

```
var average = (15 + 23 + 39) /3 ; // 25.666666666
```

2. To limit decimal places to specified number you can call toFixed() function on number and round last digit

```
var avg = average.toFixed(3);// returns 25.667
```

Arrays

Arrays

1. JavaScript arrays are used to store multiple values in a single variable.
2. An array is used to store a collection of data
3. It is an ordered collection which store elements in sequence
4. We can use array to store list of something like:
    a. Students
    b. Cars
    c. Food items

Arrays

1. If you want to store temperature of last 7 days in variable, then you have to create 7 variables

```
var mondayTemprature = 23;
var tuesdayTemprature = 12;
var wednesdayTemprature = 35;
var thursdayTemprature = 30;
var fridayTemprature = 27;
var saturdayTemprature = 19;
var sundayTemprature = 22;
```

Arrays

1. Now what if you want to store temperature of morning and evening for 7 days. That will be 14 variables
2. How about temperature for a year morning and evening that 730
3. It will be very difficult to manage and assign names to these variables

Arrays

1. If you want to find out all days temperature was above 30 then it will be quite difficult
2. If you want to sort temperature in ascending or descending order that will not be possible with separate multiple variables

Arrays

1. With Arrays you can create single variable and hold all temprature in it.
2. With Array you will be able to find and sort temperature easily

```
var temperatures = [34,12,27,65,34,28,19];
```

Creating an Arrays

1. Creating an Array using array literal

```
var food = ["Pizza",   "Burger",   "Snacks"];
```

2. Creating an Array using ***new*** Keyword

```
var foods = new Array("Pizza", "Burger","Snacks");
```

3. Both are same, first one more recommended way to create array

Creating an Arrays

1. Array can be created for all datatypes or you can mix them in single array

```
var   arr1   =  ["Hello","World",  "Bye"];
var   arr2   =  [29,  38,16,22];
var   arr3   =  [true,false,true,false,false];
 var   arr4   =  [23.2,45.8,  98.12];
 var   arr5   =  [{name: "John"}, {name: "Jason"}];
 var   arr6   =  [74,  "Hello",  true, {name: "John"}];
```

Accessing Array Elements

1. You access an array element by referring to the index number.
2. To access element you provided number in square brackets

```
var foods = ["Pizza", "Burger", "Snacks"];
foods[0];  // Pizza
foods[1];  // Burger
foods[2];  // Snacks
```

Accessing Array Elements

1. Store result in variable or show output directly

```
var foods = ["Pizza", "Burger", "Snacks"];
var a = foods[0];          // Pizza
var b = foods[1];          // Burger
var c = foods[2];          // Snacks
alert(a);                  // Pizza
alert(foods[2]);           // Snacks
```

Accessing Array Elements

1. Range of array index is from 0 to Array's length - 1
2. First element is on index 0
3. Second element on index 1
4. Third on index 2
5. Last element will be on array's length -1 e.g
    a. Array has 5 element then last element in on 4th index
    b. Array has 8 element then last element in on 7th index

Accessing full Array

1. The full array can be accessed by referring to the array name

```
var foods = ["Pizza", "Burger", "Snacks"];
console.log(foods); // Pizza, Burger, Snacks
```

Accessing Index that does not exists

1. If you create array with 3 elements and try to access 4th element, it will return *undefined*
2. There will be no error in accessing index that does not exists

```
var foods = ["Pizza", "Burger", "Snacks"];

console.log(foods[2]); // Snacks
console.log(foods[3]); // undefined
console.log(foods[8]); // undefined
```

Add/Update Element using index

1. You can use array index to add or update elements in array

```
var foods  = [];
foods[0]   = "Pizza";
foods[1]   = "Burger"
foods[2]   = "Snacks";

console.log(foods[0]); //Pizza
console.log(foods[2]); //Snacks
```

Add/Update Element using index

```
var foods = ["Pizza", "Burger", "Snacks"];

console.log(foods[1]); // Burger
foods[1] = "Sandwich"; // Updating existing element
console.log(foods[1]); // Sandwich
foods[3] = "French Fries"; // Adding 1 more element
console.log(foods[3]); // French Fries
```

Length property

1. You can find out number of elements in an array by length property

```
var foods = ["Pizza", "Burger", "Snacks"];
console.log(foods.length); // 3


var arr = [];
console.log(arr.length); // 0
```

Push function

1. Push function lets you add element in array without worrying about index
2. You don’t need to remember last index used to add in element, just call push function on array

```
var foods = [];
foods.push("Pizza");
foods.push("Burger");
foods.push("Snacks");
```

Push function

```
var foods = [];
foods.push("Pizza");
foods.push("Burger");
foods.push("Snacks");

alert(foods[0]);    // Pizza
alert(foods[1]);    // Burger
alert(foods[2]);    // Snacks
```

Push function -- Multiple input

```
var foods = [];
foods.push("Pizza");

foods.push("Burger", "Snacks");// Will add in sequence
foods.push("Sandwich");
```

```
alert(foods[0]);     // Pizza
alert(foods[1]);     // Burger
alert(foods[2]);     // Snacks
alert(foods[3]);     // Sandwich
```

Array Data Structure

1. A data structure is a specialized format for organizing, processing, retrieving and storing data
2. It enables efficient access and modification of data.
3. You can use same array syntax as:
    a. Random Access
    b. Stack (Last in First out)
    c. Queue (First in First out)

Random Access

1. You can access array elements from any index and update them

```
var foods = ["Pizza", "Burger", "Snacks"];

console.log(foods[1]); // Burger
foods[1] = "Sandwich"; // Updating existing element
console.log(foods[1]); // Sandwich
```

Stack

Last in First out

(LIFO)

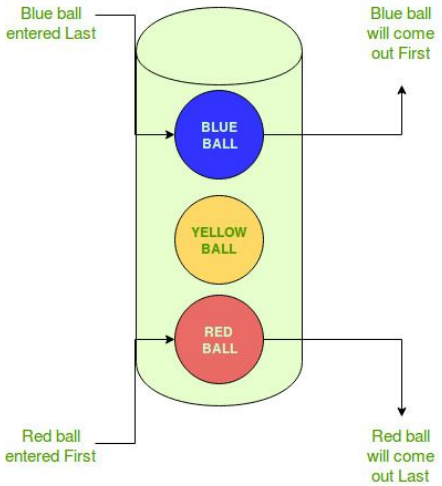
Stack (Last in First out)

1. Stack is a linear data structure represented by a real physical stack or pile where insertion and deletion of items takes place at one end called top of the stack.
2. The basic concept can be illustrated by thinking of your data set as a stack of plates or books where you can only take the top item off the stack in order to remove things from it
3. The basic implementation of a stack is also called a LIFO (Last In First Out)

Stack (Last in First out)

LIFO is an abbreviation for **last in, first out**. It is a method for handling data structures where the **first element** is processed last and the **last element** is processed first.

Real-life example:’



Stack (Last in First out)

1. To behave as stack you have to use push and pop function on array
2. **Push** function will add element at the **end** in array
3. **Pop** function will remove and return **last** elements from array

Stack (Last in First out)

```
var foods = [];
foods.push("Pizza");
foods.push("Burger");
foods.push("Snacks");

console.log("Length "+foods.length); // Length 3
console.log(foods.pop()); // Remove Snacks from array
console.log(foods.pop()); // Remove Burger from array
console.log("Length "+foods.length); // Length 1
```

Queue

First in First out

(FIFO)

Queue (First in First out)

1. Queue is a linear data structure represented by a real physical queue.
2. You can think of it as a line in a grocery store or banks
3. A queue is open at both its ends.
4. The basic implementation of a stack is also called a FIFO (First-In-First-Out)

Queue (First in First out)

1. To behave as queue you have to use push and shift function on array
2. **Push** function will add element at the **end** in array
3. **Shift** function will remove and return **first** elements from array

Queue (First in First out)

```
var foods = [];
foods.push("Pizza");
foods.push("Burger");
foods.push("Snacks");

console.log("Length "+foods.length); // Length 3
console.log(foods.shift());// Remove Pizza from array
console.log(foods.shift());// Remove Burger from array
console.log("Length "+foods.length); // Length 1
```

Unshift function

1. To add element in array we use push function or index
2. These will add element at end of array
3. If you want to add element at the start of array and move all element one index ahead then we can use unshift function

Unshift function

```
var foods = [];

foods.push("Pizza");
foods.push("Burger");
foods.push("Snacks");

console.log("Length "+foods.length); // Length 3

console.log(foods[0]); // "Pizza"  foods.unshift("Sandwich");

console.log(foods[0]); // "Sandwich"

console.log(foods[1]); // "Pizza"
```

Comparison LIFO Vs FIFO

FIFO	LIFO
It stands for First-In-First-Out approach in programming.	It stands for Last-In-First-Out approach in programming.
In this, the new element is inserted below the existing element, So that the oldest element can be at the top and taken out first.	In this, the new element is inserted above the existing element, So that the newest element can be at the top and taken out first.
Therefore, the first element to be entered in this approach, gets out First.	Therefore, the first element to be entered in this approach, gets out Last.
In computing, FIFO approach is used as an operating system algorithm, which gives every process CPU time in the order they arrive.	In computing, LIFO approach is used as a queuing theory that refers to the way items are stored in types of data structures.
The data structure that implements FIFO is Queue.	The data structure that implements LIFO is Stack.

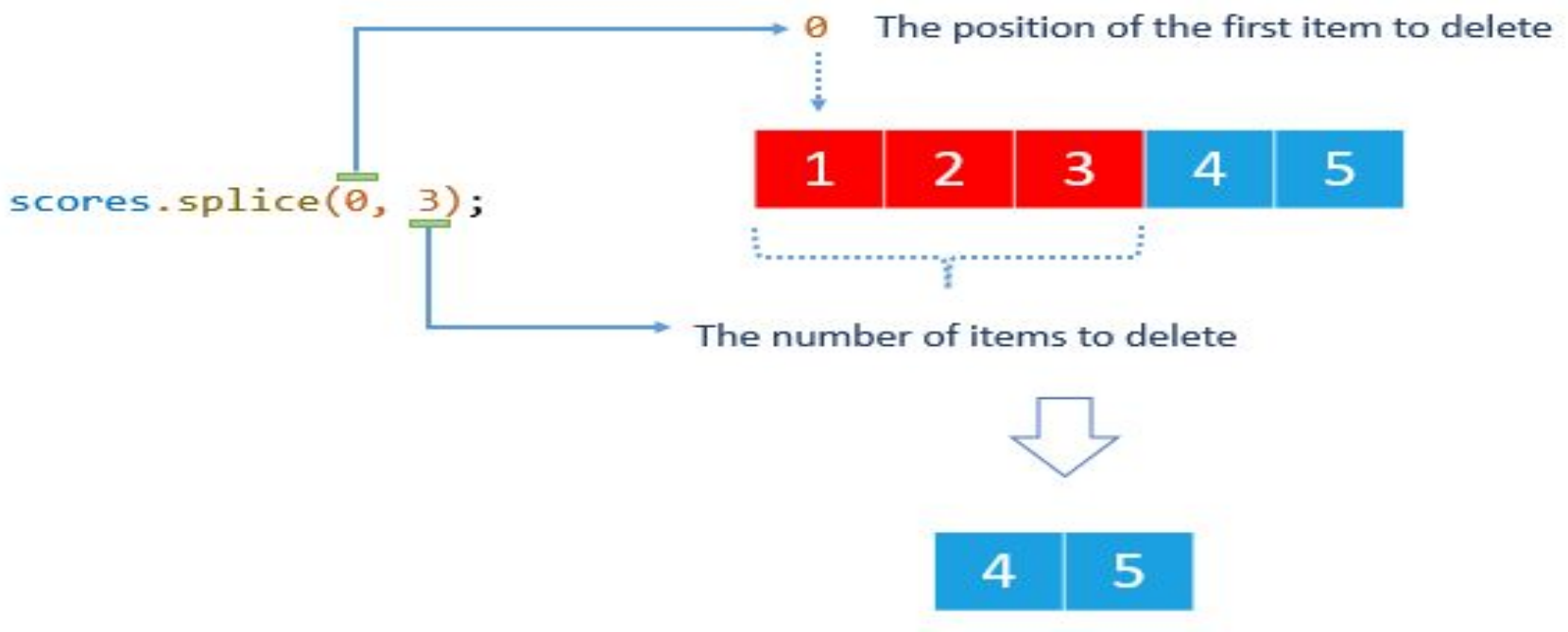
Splice function

1. To add element in array we have used
    a. Push function -- add element in last
    b. Unshift function -- add element in start
    c. Index -- add element in last or replace existing element
2. If we want to add element in middle of array or any index other than first/last then we can use splice function

Splice function

1. Splice function can add one or more element on particular index in array
2. Splice function can replace one or more element on particular index in array
3. Splice function returns elements which removed from array, if no element removed then returns empty array

Splice function



Splice function

The position specifies the position of the first item to delete and the num argument determines the number of elements to delete. The splice() method changes the original array and returns an array that contains the deleted elements.

```
let months = ["January", "February", "Monday", "Tuesday"];
let days = months.splice(2);
console.log(days); // ["Monday", "Tuesday"]
```

Splice function

```
var foods = ["Pizza", "Burger", "Snacks"];
console.log(foods);// "Pizza","Burger","Snacks"
foods.splice(1,0,"Sandwich");
console.log(foods);

//"Pizza","Sandwich", "Burger","Snacks"
```

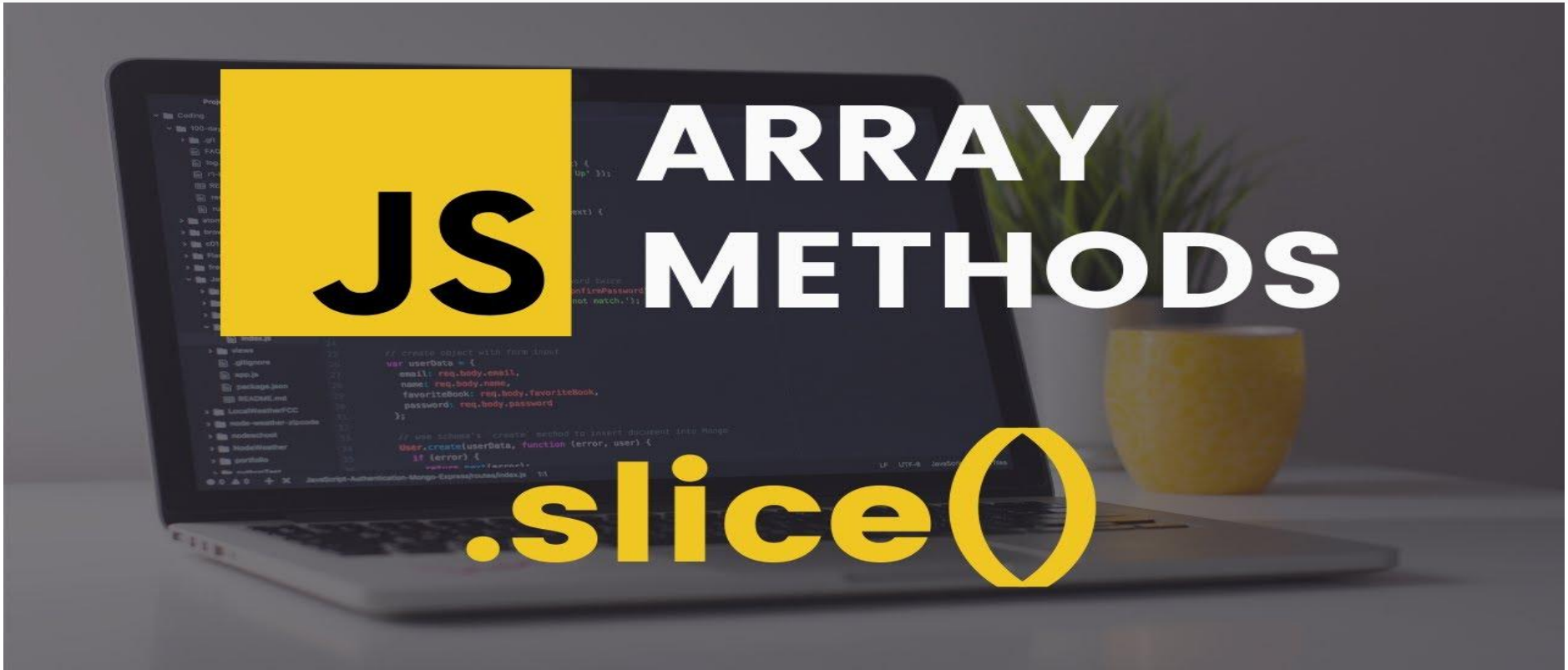
This will add 1 element on index 1 and move all elements one index forward

Splice function

```
var foods = ["Pizza", "Burger", "Snacks"];
console.log(foods);// "Pizza","Burger","Snacks"
foods.splice(1,0,"Sandwich", "Fries");
console.log(foods);

//"Pizza","Sandwich","Fries","Burger","Snacks"
```

This will add 2 element on index 1 and 2 and move all elements two index forward


JS
ARRAY
METHODS
.slice()

Slice function

1. To create array from element of existing array you can use slice function
2. We can create subset of array from existing array
3. Slice takes start and end index of array to create new array
4. Slice Syntax:
    a. slice(index of array, end index)
    b. End index is exclusive, if you say 4 that means 3rd index

Slice function

```
var foods = ["Pizza", "Burger", "Snacks", "Sandwich",
"Fries"];
console.log(foods);//
"Pizza","Burger","Snacks","Sandwich","Fries"
var arr = foods.slice(1,3);
console.log(foods);// output same as above
console.log(arr); //"Burger","Snacks"
```

Thanks

End of Module-1 (Lecture-6)