

ETHEREUM 2.0 MASTERY PROGRAM

Instructor: Raja Rizwan Saleem



Module

ONE

JAVASCRIPT CRASH COURSE

Instructor: Raja Rizwan Saleem



SESSION-5



Break

```
for (var i = 0; i < 8;
    i++){    if(i == 4) {
                break;
            }
            console.log("i = "+i);
        }
```

Output

i = 0

i = 1

i = 2

i = 3

Continue

```
for (var i = 0; i < 8; i++){  
    if(i == 4) {  
        continue;  
    }  
    console.log("I = "+i);  
}
```

Output

I = 0

I = 1

I = 2

I = 3

I = 5

I = 6

I = 7

Nested Loops

1. If a loop exists inside the body of another loop, it's called nested loop.

```
for (var i = 0; i < 3; i++){  
    for(var j = 0; j < 2; j++) {  
        console.log("I = "+i+" J = "+j);  
    }  
}
```

Output:

```
I = 0 J = 0 I  
= 0 J = 1 I  
= 1 J = 0 I  
= 1 J = 1 I  
= 2 J = 0 I  
= 2 J = 1
```

Nested Loops

```
let a = ["a", "e", "i", "o", "u", "A", "E", "I", "O", "U"];
let input = prompt("please enter any character"); let check = false; for
(let i = 0; i < a.length; i++) {

if (input === a[i]) {

check = true;

console.log("It is a vowel"); break;
}
} if (check === false) {

console.log("it is a consonant"); }
```

Task

1. Find out if number is prime number or not

Prime number is divisible only by itself and 1 (e.g. 2, 3, 5, 7, 11).

Tasks

1. Check if number is even or odd using parseInt
2. Check number is positive or negative
3. Check an alphabet is a vowel or not

Task

1. Generate triangle output like below, Hint: nested loop required

```
  *
 * *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

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 mondayTemperature = 23;  
var tuesdayTemperature = 12;  
var wednesdayTemperature = 35;  
var thursdayTemperature = 30;  
var fridayTemperature = 27;  
var saturdayTemperature = 19;  
var sundayTemperature = 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];  
var b = foods[1]; var c = foods[2]; alert(a);  
alert(foods[2]);
```

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 exist

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

```
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)

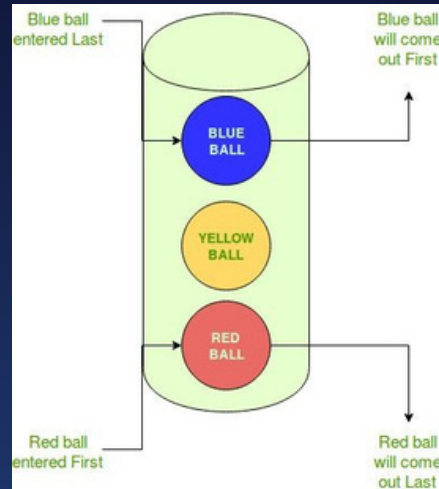
Stack (Last in First out)

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
The basic implementation of a stack is also called a
3. 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.

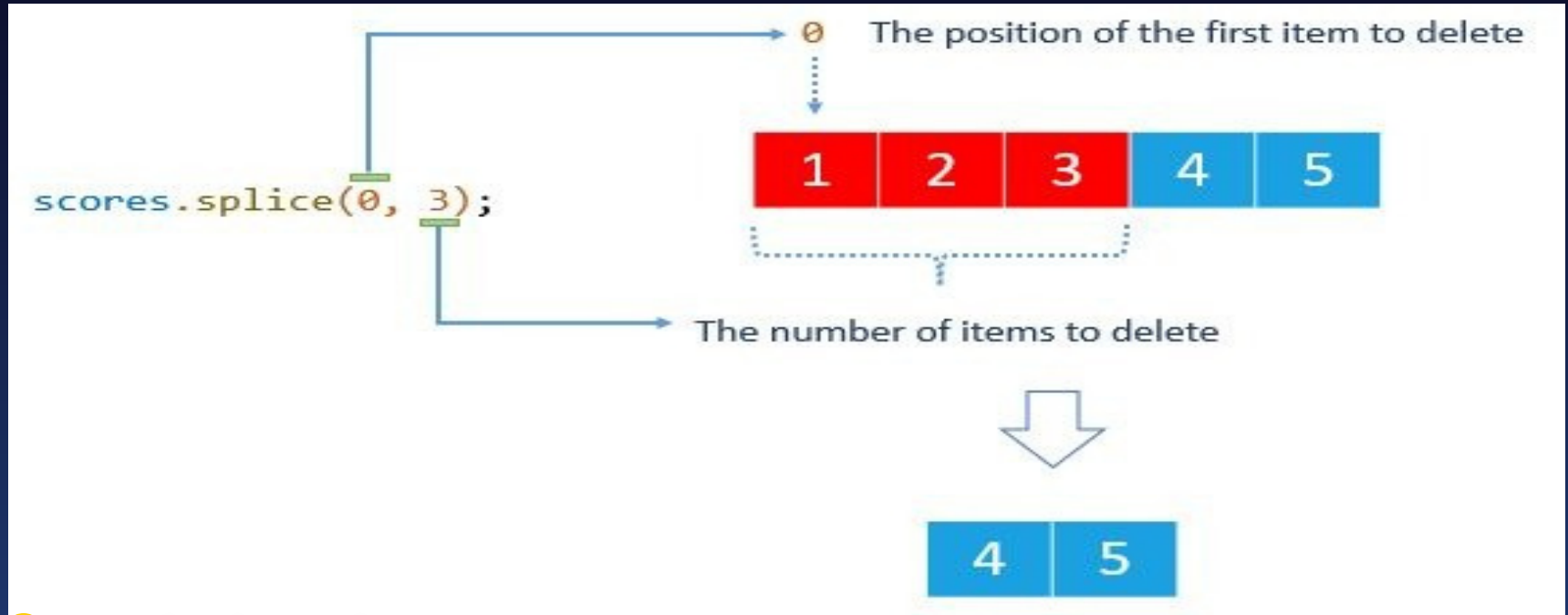
Splice function

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

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"
```

THANK-YOU

