JAVASCRIPT

JavaScript is the world's most popular programming language.

JavaScript is the programming language of the Web.

JavaScript is easy to learn.

JavaScript is the control of the Web Page.

Computer Memory

- **Temporary Memory**
  All the data on the memory has lost when the electrictiy is off or the computer is switched off.
  eg - RAM (Random Access Memory)
- **Permanent Memory**
  All the data on the memory retain when the electrictiy is off or the computer is switched off.
  eg - CD, Memory Stick, Hard Disk, SSD

Variable

Variable is a location or storage on the computer memory (RAM).
Variable can keep single value.

Variable Declaration

```
let a; //declaration
let b; //declaration
```

Variable Assignment

```
a = 10; //assignment
b = 20; //assignment
```

Variable Declaration And Assignment

```
let c = 100; //declaration and assignment
let d = 200; //declaration and assignment
```

Variable Declaration Rules

- should not start with capital letter. (eg. let Num1;)
- should not start with number. (eg. let 1num1;)
- should not use special character. (eg. let num^@1;)
- should not use space. (eg. let number of person;)
- should start with small letter. (eg. let num1;)
- should use _ or camel case instead of space.
  (eg. let numberOfPerson;[camel case])
  (eg. let number_of_person;[_])
- must be meaningful;

How to store the variable in memory (background)

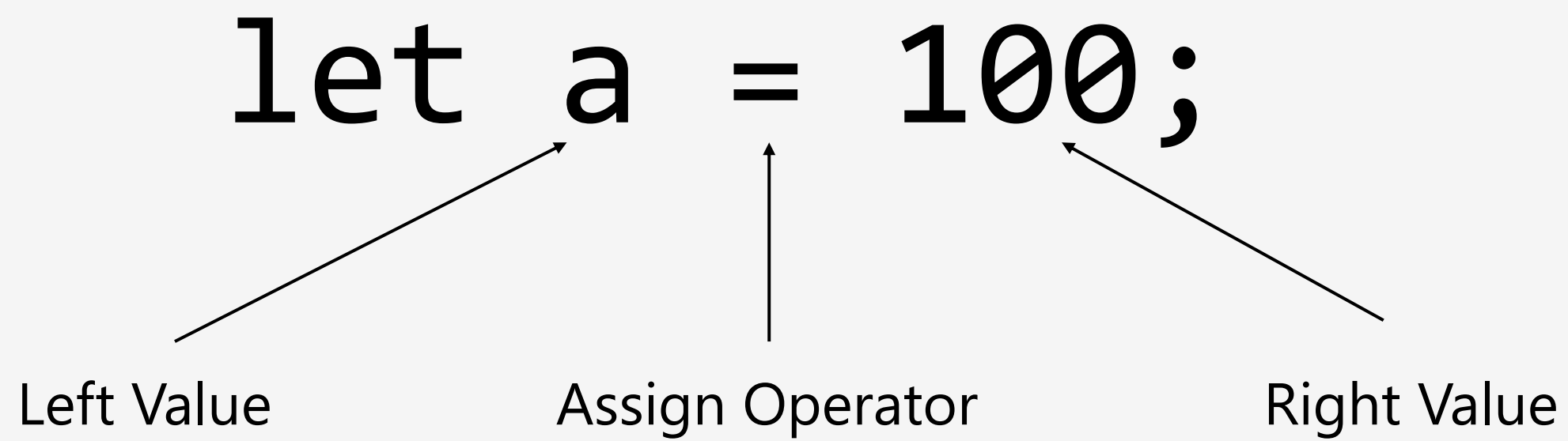
```
let a; //step1 (declaration)
let b; //step1 (declaration)

a = 10; //step2 (assignment)
b = 20; //step2 (assignment)

let c = 100; //step1 declaration & assignment
c = 200; //step2 assignment
```

Example	MEMORY (RAM)	
	ADDRESS	VALUE
a (step1)	&a (0x0000)	NULL
b (step1)	&b (0x0001)	NULL
a (step2)	&a (0x0000)	10
b (step2)	&b (0x0001)	20
&c (step1)	&c (0x0002)	100
&c (step2)	&c (0x0002)	200

Working Flow Of The Assign Operator (=)



Note - Assign operator assign the right value to the left value;

Data Types

Primitive Data Types

- Boolean (true,false)

```
let status = true;
```

- Null (no value)

```
let value = null;
```

- Undefined (a declared variable but hasn’t been given a value)

```
let value = undefined;
```

- Number (integer,float,etc ..)

```
let value = 13;
let value2 = 3.14;
```

- String (an array of characters i.e words)

```
let name = “Aung Aung”;
```

Object Data types

- Object

```
let mgmg = {
        name: “mg mg”,
        age: 17,
        address: “Yangon”
};
```

- Array

```
let tempArray = [
        “mg mg”,
        1.5,
        100,
        true,
        [1,2,3],
        {
           name: “banana”,
           color: “yellow”
        }
];
```

Array

- array can keep multiple values.
- you can call its value by it’s index or it’s keys.
- [] represent the array
- array index start from 0;

Eg –

```
// array declaration and assignment
let arr = [1,2,3,100,200,300];

// call the array with index number
console.log(arr[3]); //100

// assignment the value to the array with index number
arr[2] = 500; // [1,2,500,100,200,300];
console.log(arr);
```

//calling array note
Index Array ဆိုရင် index နဲ့ခေါ်
Key Value Array ဆိုရင် Key နဲ့ခေါ်

Object

Object is everything in the real world.
There are two sectors in object.

- Properties
- Behaviors

Convert To Computerise
Object is object.

- Properties => Variable
- Behaviors => Function

Building Direct Object
Advantage

- can create object directly.

Disadvantage

- not symmetric

Building Template Object
Advantage

- symmetric

Disadvantage

- To be symmetric, need to create template class and need to create object from template class.

Object Example

Building Direct Object

```
let mgmg = {
            name: "mg mg",
            age: 15,
            profession: 'student',

            study: function(){
                console.log('Studying');
            }
        }
```

Building Template Class

```
 class Person{
        name;
        age;
        address;
        study() {
            console.log('studying');
        }
        eat() {
            console.log('Eat');
        }
 }
```

Building Object From Template Class

```
 let mm = new Person;
 mm.name = 'MGMG';
 mm.age = 32;
 mm.address = 'yangon';
 mm.eat();
 console.log(mm);
```

Operators (1)

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Function

- Function ကိုလုပ်ငန်းစဉ်လို့ခေါ်တယ်။ မိမိလုပ်ချင်သော လုပ်ငန်းစဉ်တွေကို Function ထဲမှာ သွားရေးရပါမယ်။
- Function သည် reuse ပြန်လုပ်လို့ရသည်။
- Function မှာ အပိုင်းနှစ်ပိုင်းရှိတယ်
  - Function Call
  - Function Declaration

  Note – Function များသည် Function Call မှသာ အလုပ်လုပ်သည်။

Example

```
//function declaration
function a(){
        console.log(“my first function”);
}

//function call
a();
```

Predefined & User Defined Function

Predefined Function

အလွယ်တကူ အဆင်သင့် ခေါ်ယူအသုံးပြုနိုင်အောင် ကြိုတင် သတ်မှတ်ထားသော Function များ ဖြစ်သည်။

Example –

```
alert('my first predefined function');
console.log('my first predefined function');
```

User Defined Function

ကိုယ်တိုင်သတ်မှာရေးရသော Function များ ဖြစ်သည်။

Example –

```
//function declaration
function addTwoNumber(){
    let num1 = 10;
    let num2 = 20;
    let result = num1 + num2;
    console.log(result);
}

//function call
addTwoNumber();
```

Function Argument & Parameter

Function တွင်

- No Argument Function
- Argument Function ဟူ၍ နှစ်မျိုးရှိသည်။

Argument နှင့် Parameter သည် အတူတူပင်ဖြစ်သည်။ ထားသိုသော နေရာကွဲ၍ အခေါ်အဝေါ်ကွဲသွားသည်။

- Function Declaration တွင် Parameter လို့ခေါ်ပြီး
- Function Call တွင် Argument (the real value received by the function) လို့ခေါ်သည်။

No Argument Function

```
function addTwoNumber(){
        let num1 = 10;
        let num2 = 20;
        console.log(num1+num2);
}
addTwoNumber(); //result is 30
```

No Parameter ()

No Arguments ()

Argument Function

```
function addTwoNumber(num1,num2){
        console.log(num1+num2);
}
addTwoNumber(50,100); //result is 150
```

Two Parameter(num1,num2)

Two argument(50,100)

Return & No Return Function

No Return Function

No Return Function များသည် Function အတွင်းသာ လုပ်ငန်းစဉ် ပြီးမြောက်သွားကြသည်။ တစ်ခြားသော အရာများနှင့် တွဲဖက်လုပ်ဆောင်၍ မရ။

```
function addingTwoNumber(num1,num2){
    console.log(num1+num2);
}
addingTwoNumber(20,40); // 60
addingTwoNumber(10,30); // 40
```

Return Function

Return Function များသည် တန်ဖိုးတစ်ခုကို return(ပြန်ထုတ်) ပြန်သည်။
ထို function သည် return ပြန်သော တန်ဖိုးရှိသည်။
next process များကို ဆက်လုပ်ရန် တစ်ခြားသော အရာများနှင့် တွဲဖက်လုပ်ဆောင်ရာတွင် ရေးလေ့ရေထရှိသည်။

```
function addingTwoNumber(num1,num2){
    return num1+num2;
}

let result = addingTwoNumber(50,60) + 100; // 210
```

Loop Explanation

```
i++ (i = i+ 1)
i-- (i = i -1)
```

```
for (let i = 0; i < 5; i++) {

      console.log('hello' + (i+1));

}
```

1) let i =0 (initialization)
2) i< 5 (condition)
3) i++ (increment /decrement)

first loop
-> initialize
-> condition (true)
-> process the code in the scope

greater than second loop and second loop
-> increment
-> condition (true)
-> process the code in the scope

Note - loop break when the condition is false.

1 loop
i =0 (initialize)
true (condition)
hello 1 (result)

2 loop
i = 1 (i++)
true (condition)
hello 2 (result)

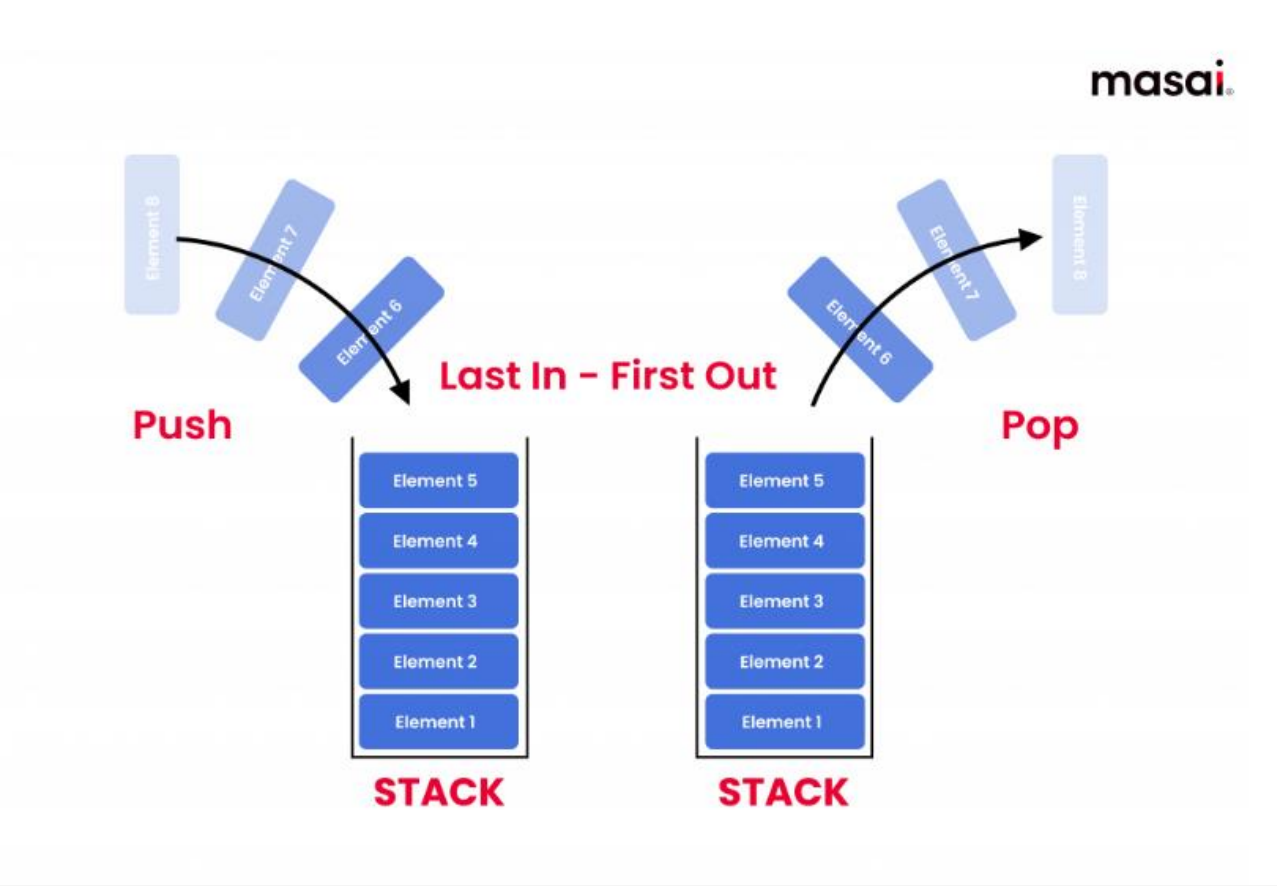
3 loop
i = 2 (i++)
true (condition)
hello 3 (result)

4 loop
i = 3 (i++)
true (condition)
hello 4 (result)

5 loop
i = 4 (i++)
true (condition)
hello 5 (result)

6 loop
i = 5 (i++)
false (condition)
loop break

Stack (Last In First Out) - 1



Stack (Last In First Out) - 2

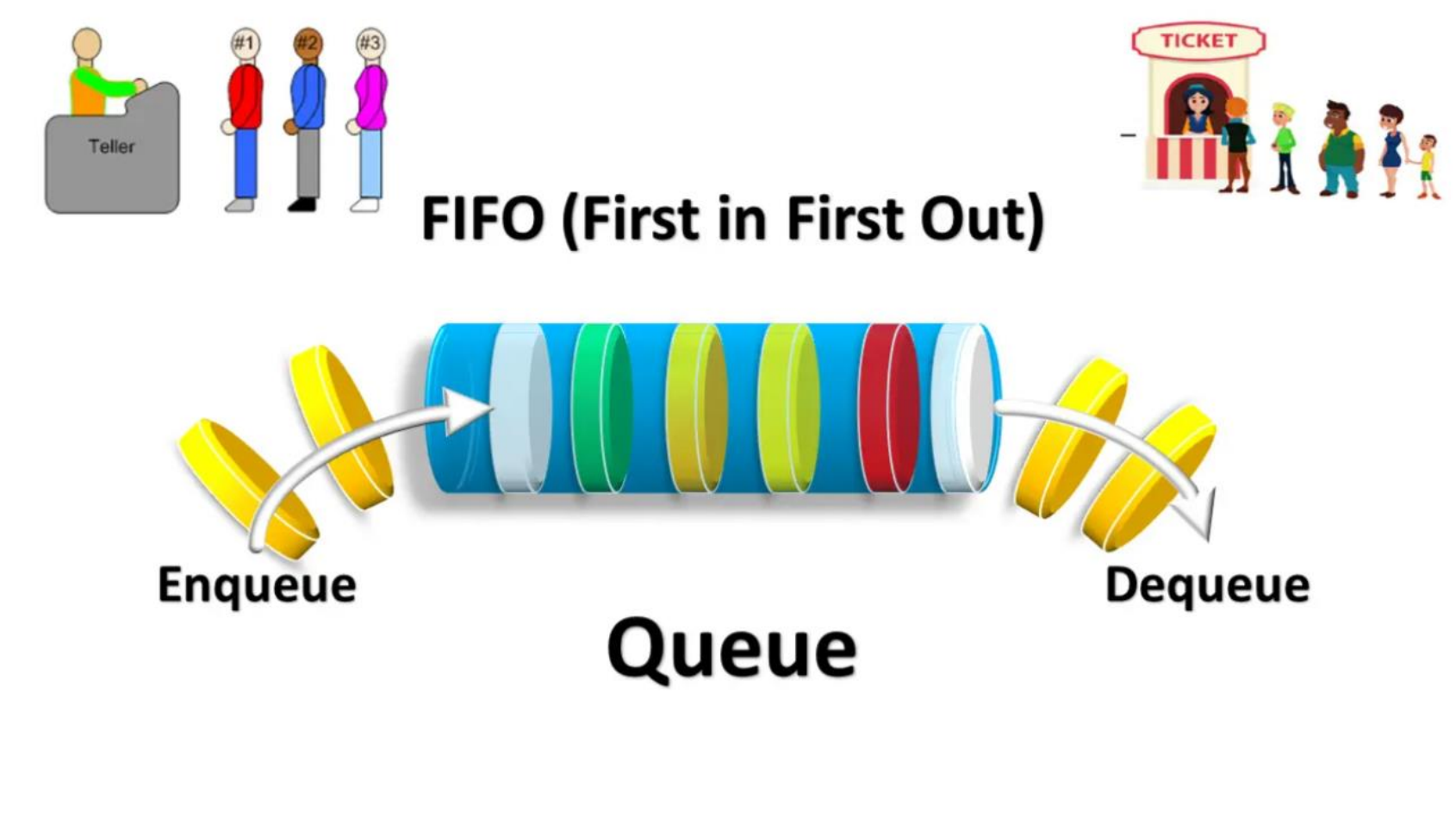
Using Stack In Javascript

```
const hobbies = [‘Sports’, ‘Cooking’];

//add values to array
hobbies.push(‘playing games’);
hobbies.push(‘swimming’);
console.log(hobbies);

//remove values from array
hobbies.pop()
hobbies.pop()
console.log(hobbies);
```

Queue (First In First Out) - 1



Queue (First In First Out) - 2

Using Stack In Javascript

```
const people = [‘Aung Aung’,’Kyaw Kyaw’];

//add values to array
people.push(‘Min Min’);
people.push(‘Ma Ma’);
console.log(people);

//remove values from array
people.shift()
people.shift()
console.log(people);
```

Rest & Spread Operator

What is rest operator?

A rest operator is a type of parameter that gets all of the remaining parameters of a function call via an Array. It enables us to handle a variety of inputs as parameters in a function.
Example –

```
const toArray = (...args) => {
    return args;
}
console.log(toArray(1,2,3,4,5));
```

What is spread operator?

The spread operator divides an array or object into separate elements or properties. The spread operator is mostly used if you want to duplicate the content of an array or an object.
Example –

```
const hobbies = ['sports','playing music'];
const copiedHobbies = [...hobbies];
copiedHobbies.push('cooking');
```

Destructing

What is destructing?

Destructuring Assignment is a JavaScript expression that allows to unpack of values from arrays, or properties from objects, into distinct variables data can be extracted from arrays, objects, and nested objects, and assigned to variables.

Note- Destructing does not change the original object.

Example –

```
// Create an Object
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50
};

// Destructuring
let {firstName, lastName} = person;
```

Promise

What is promise?

"I Promise a Result!"

"Producing code" is code that can take some time

"Consuming code" is code that must wait for the result

A Promise is an Object that links Producing code and Consuming code

Example –

```
// Create a Promise
const fetchData = () => {
    const promise = new Promise((resolve,reject)=>{
        setTimeout(() => {
            resolve('hello');
        }, 1000);
    })
    return promise;
}

// Using a Promise
fetchData()
.then( value => console.log(value))
.then( value => 'hello1')
.then( value => 'hello2')
.catch( err => console.log(err) );
```