Javascript Basic Notes

*Keyword

কীওয়ার্ড হল সংরক্ষিত শব্দ যা প্রোগ্রামিং ভাষার সিনট্যাক্সের অংশ।

Here is the list of keywords available in JavaScript: Let, var, const, if, else, switch, break, case, for,p await, catch, class, continue, debugger, default, delete, Do, enum, export, extends, False, finally, function, Implements, import, in, instanceof, interface, new, null, package, private, Protected, public, return, super, Static, this, throw, try, true, Typeof, void, while, with, yield.

Let:

লেট ডিক্লেয়ারেশন একটি ব্লক-স্কোপড লোকাল ভেরিয়েবল ঘোষণা করে, ঐচ্ছিকভাবে এটিকে একটি মানের শুরু করে।

Example:
```
let a = 12;
let b = 12;
console.log(a+b);
```
Output: 24

Var:

var বিবৃতি একটি ফাংশন-স্কোপড বা বিশ্বব্যাপী-স্কোপড ভেরিয়েবল ঘোষণা করে, ঐচ্ছিকভাবে এটিকে একটি মানের শুরু করে।

Example:
```
var a = 12;
var b = 12;
console.log(a+b);
```
Output: 24

Const:

সর্বদা const সহ একটি ভেরিয়েবল ঘোষণা করুন যখন আপনি জানেন যে মান পরিবর্তন করা উচিত নয়।

Example:
```
const PI = 3.141592653589793;
PI = 3.14;
```

*Variable

পরিবর্তনশীল মানে এমন কিছু যা পরিবর্তন করতে পারে। জাভাস্ক্রিপ্টে, একটি ভেরিয়েবল ডেটা মান সংরক্ষণ করে যা পরে পরিবর্তন করা যেতে পারে।

4 Ways to Declare a JavaScript Variable:
Using **var**, Using **let**, Using **const**, Using **nothing**.

Example1: let gettingName = 'Hello';
console.log(gettingName + ' World!');
Output: Hello World!

Example2: var x = 5;
var y = 6;
var z = x + y;
console.log(z);
Output: 11

Example3: const PI = 3.141592653589793;
Output: 3.141592653589793;

*Data Types (Basic)

ডেটা টাইপগুলি মূলত নির্দিষ্ট করে যে কোনও প্রোগ্রামের মধ্যে কী ধরণের ডেটা সংরক্ষণ করা যায় এবং ম্যানিপুলেট করা যায়।

The five most basic types of data in Javascript**: Strings**, **Numbers**, **Booleans**, **Undefined**, and **Null**.

Number:

সংখ্যা হল এমন মান যা গাণিতিক ক্রিয়াকলাপে ব্যবহার করা যেতে পারে। সংখ্যার জন্য আপনার কোন বিশেষ সিনট্যাক্সের প্রয়োজন নেই — শুধু সরাসরি জাভাস্ক্রিপ্টে লিখুন।

Example: let num = 12;
let num = 12.121212;

String:

জাভাস্ক্রিপ্টে, স্ট্রিং হল টেক্সট দিয়ে তৈরি মান এবং এতে অক্ষর, সংখ্যা, চিহ্ন, বিরাম চিহ্ন এবং এমনকি ইমোজিও থাকতে পারে!

Example: "It's six o'clock.";
'Remember to say "please" and "thank you."';

Boolean:

জাভাস্ক্রিপ্টে, একটি বুলিয়ান মান হল যেটি “Yes” or “False” হতে পারে। আপনি যদি কিছু সম্পর্কে "হ্যাঁ" বা "না" জানতে চান, তাহলে আপনি বুলিয়ান ফাংশনটি ব্যবহার করতে চান। এটি অত্যন্ত সহজ শোনাচ্ছে, তবে জাভাস্ক্রিপ্ট প্রোগ্রামিংয়ে বুলিয়ানগুলি সর্বদা ব্যবহৃত হয় এবং সেগুলি অত্যন্ত দরকারী। "On" or "Off", "Yes" or "No", "True" Or "False", অথবা যার শুধু একটি অস্থায়ী উদ্দেশ্য থাকা প্রয়োজন, তা সাধারণত বুলিয়ানদের জন্য উপযুক্ত।

Example:
```
var openDoor = false;
openDoor = true;
console.log(openDoor);
```
Output: true

Undefined:

অনির্ধারিত property নির্দেশ করে যে একটি ভেরিয়েবলের একটি মান বরাদ্দ করা হয়নি, বা ঘোষণা করা হয়নি। পরে চাইলে আবার এই ভেরিয়েবলের মধ্যে মান সেট করা যাবে।

Example:
```
var name;
console.log(name);
```
Output: undefined

Example:
```
var name;
name = "Rajib Rahman"
console.log(name);
```
Output: Rajib Rahman

Null:

মান নাল কোন বস্তুর মান ইচ্ছাকৃত অনুপস্থিতি প্রতিনিধিত্ব করে. এটি জাভাস্ক্রিপ্টের Primitive মানগুলির মধ্যে একটি এবং বুলিয়ান ক্রিয়াকলাপের জন্য এটিকে মিথ্যা বলে গণ্য করা হয়।

Example:
```
var myRank = null;
console.log(myRank);
```
Output: null

*Javascript Type Conversion

Javascript Conversion এর মাধ্যমে জাভাস্ক্রিপ্ট ভেরিয়েবলগুলিকে একটি নতুন ভেরিয়েবল এবং অন্য ডেটা টাইপে রূপান্তর করা যেতে পারে।

There are **two types** of type conversion in JavaScript**:**
1. **Implicit Conversion -** automatic type conversion.
2. **Explicit Conversion -** manual type conversion.

JavaScript Implicit Conversion:

কিছু পরিস্থিতিতে, জাভাস্ক্রিপ্ট স্বয়ংক্রিয়ভাবে একটি ডেটা টাইপকে অন্য ডেটা টাইপে (সঠিক টাইপে) রূপান্তর করে। এটি Implicit Conversion হিসাবে পরিচিত।

Example1: // CONVERTING TO STRING

```
result = '3' + 2;
console.log(result) // "32"

result = '3' + true;
console.log(result); // "3true"

result = '3' + undefined;
console.log(result); // "3undefined"

result = '3' + null;
console.log(result); // "3null"
```

Remember: যখন একটি সংখ্যা একটি স্ট্রিংয়ে যোগ করা হয়, তখন জাভাস্ক্রিপ্ট সংখ্যাটিকে সংযোজন করার আগে একটি স্ট্রিংয়ে রূপান্তর করে নেয়।

Example2: // CONVERTING TO NUMBER

```
result = '4' - '2';
console.log(result); // 2

result = '4' - 2;
console.log(result); // 2

result = '4' * 2;
console.log(result); // 8

result = '4' / 2;
console.log(result); // 2
```

Example3: // BOOLEAN CONVERTING TO NUMBER

```
result = '4' - true
console.log(result); // 3

result = 4 + false;
console.log(result); // 4
```

Example4: // NULL CONVERTING TO NUMBER

```
result = 4 + null;
console.log(result);  // 4

number = 5 / null;
console.log(number);  // Infinity

result = 4 - null;
console.log(result);  // 4
```

JavaScript Explicit Conversion:

প্রয়োজন অনুসারে একটি ডেটা টাইপ থেকে অন্যটিতে রূপান্তর করা যায়। ম্যানুয়ালি যে type conversion করা হয় তা Explicit type conversion হিসেবে পরিচিত।

Convert to Number Explicitly:

Numeric strings এবং Boolean মানকে সংখ্যায় রূপান্তর করতে, **Number()** ব্যবহার করা হয়।

Example: // STRING TO NUMBER

```
result = Number('324');
console.log(result); // 324

result = Number('324e-1')
console.log(result); // 32.4

// BOOLEAN TO NUMBER
result = Number(true);
console.log(result); // 1
```

Convert to String Explicitly:

অন্যান্য ডেটা প্রকারগুলিকে স্ট্রিং-এ রূপান্তর করতে, **String()** বা **toString()** ব্যবহার করা

হয়।

Example: //NUMBER TO STRING

```
result = String(2 + 4);
console.log(result); // "6"


result = String(2 * 4);
console.log(result); // "8"

//OTHER DATA TYPE TO STRING
result = String(null);
console.log(result); // "null"

result = String(undefined);
console.log(result); // "undefined"

result = String(NaN);
console.log(result); // "NaN"

result = String(true);
console.log(result); // "true"

// USING toString()
result = (324).toString();
console.log(result); // "324"
```

Convert to Boolean Explicitly:

অন্যান্য ডেটা টাইপকে বুলিয়ানে রূপান্তর করতে, **Boolean()** ব্যবহার করা হয়। জাভাস্ক্রিপ্টে, undefined, null, 0, NaN, ও "", False-এ রূপান্তরিত হয়।

Example1:

```
result = Boolean('');
console.log(result); // false

result = Boolean(0);
console.log(result); // false

result = Boolean(undefined);
```

```
console.log(result); // false

result = Boolean(null);
console.log(result); // false

result = Boolean(NaN);
console.log(result); // false
```

অন্য সব মান True হয়।

Example2:
```
result = Boolean(324);
console.log(result); // true

result = Boolean('hello');
console.log(result); // true

result = Boolean(' ');
console.log(result); // true
```

JavaScript Type Conversion Table

টেবিলটি জাভাস্ক্রিপ্টে String, Number and Boolean এর বিভিন্ন মানের রূপান্তরদেখায়।

Value	String Conversion	Number Conversion	Boolean Conversion
1	"1"	1	true
0	"0"	0	false
"1"	"1"	1	true
"0"	"0"	0	true
"ten"	"ten"	NaN	true
true	"true"	1	true
false	"false"	0	false
null	"null"	0	false
undefined	"undefined"	NaN	false

''	""	0	false
' '	" "	0	true

*Javascript Number Methods

জাভাস্ক্রিপ্ট নম্বর অবজেক্ট হল একটি Primitive Wrapper Object যা সংখ্যাকে Represent করতে এবং ম্যানিপুলেট করতে ব্যবহৃত হয়। জাভাস্ক্রিপ্ট বিভিন্ন পদ্ধতি প্রদান করে যা সংখ্যার সাথে কাজ করে।

The most useful 12 **Number Methods** are**: parseInt()** Method, **toString()** Method, **toExponential()** Method, **toFixed()** Method, **toPrecision()** Method, **valueOf()** Method, **toLocalString()** Method, **parseFloat()** Method, **isInteger()** Method, **isFinite()** Method, **isSafeInteger()** Method, **isNaN()** Method.

parseInt() Method:

parseInt() Method প্রদত্ত String আর্গুমেন্ট parses করে এবং String থেকে parse করা একটি পূর্ণসংখ্যা প্রদান করে।

Example: let num = Number.parseInt("32.65");
console.log(num);
Output: 32

parseFloat() Method:

parseInt() Method প্রদত্ত স্ট্রিং আর্গুমেন্টকে Parse করে এবং স্ট্রিং থেকে Parse করা একটি Floating-Point নম্বর প্রদান করে।

Example: console.log(Number.parseFloat('25.678'));
console.log(Number.parseFloat('123ABC4'));
Output: 25.678
123

toString() Method:

toString() Method একটি স্ট্রিং আকারে প্রদত্ত সংখ্যা প্রদান করে। এই পদ্ধতিটি একটি ঐচ্ছিক প্যারামিটার হিসাবে রেডিক্স (গাণিতিক সংখ্যা পদ্ধতির ভিত্তি) গ্রহণ করে এবং নির্দিষ্ট সংখ্যা বস্তুর Representing একটি স্ট্রিং প্রদান করে।

Example: let num1 = 213;
console.log(num1.toString());

```
let num2 = -673;
console.log(num2.toString());
```

Output: 213
-673

toExponential() Method:

toExponential() Method একটি স্ট্রিং প্রদান করে যা প্রদত্ত সংখ্যার সূচকীয় স্বরলিপি উপস্থাপন করে। এই Method টি একটি ঐচ্ছিক প্যারামিটার হিসাবে ভগ্নাংশ ডিজিট গ্রহণ করে যা দশমিক বিন্দুর পরে সংখ্যার সংখ্যা নির্দিষ্ট করে।

Example:
```
let num1 = 23425;
console.log(num1.toExponential());
let num2 = 342;
console.log(num2.toExponential(2));
```

Output: 2.3425e+4
3.42e+2

toFixed() Method:

toFixed() একটি স্ট্রিং প্রদান করে, যেখানে একটি নির্দিষ্ট সংখ্যক দশমিকের সাথে লেখা সংখ্যা রয়েছে।

Example:
```
let num1 = 234.345;
console.log(num1.toFixed(1));
let num2 = -783.234;
console.log(typeof num2.toFixed(2));
```

Output: 234.3
string

toPrecision() Method:

toPrecision() একটি নির্দিষ্ট দৈর্ঘ্যের সাথে লেখা একটি সংখ্যা সহ একটি স্ট্রিং প্রদান করে এবং এটি সংখ্যার দৈর্ঘ্য নির্ধারণ করে যে, সংখ্যাটি কত ডিজিটের হবে।

Example:
```
let num1 = 234.345;
console.log(num1.toPrecision(6));
let num2 = -783.234;
console.log(num2.toPrecision(8));
```

Output: 234.345
-783.23400

valueOf() Method:

valueOf() Method একটি সংখ্যা বস্তুর Primitive মান প্রদান করে। valueOf() Method একটি সংখ্যা হিসাবে একটি সংখ্যা প্রদান করে।

Example:
```
let num1 = 234.345;
console.log(num1.valueOf());
console.log((-327).valueOf());
```
Output:
```
234.345
-327
```

toLocalString() Method:

JavaScript toLocaleString() Method একটি সংখ্যার ভাষা-সংবেদনশীল উপস্থাপনা সহ একটি স্ট্রিং প্রদান করে।

Example:
```
let num = 762359.237;
// Indian
console.log(num.toLocaleString('en-IN'));
// Chinese
console.log(num.toLocaleString('zh-Hans-CN-u-nu-hanidec'));
```
Output:
```
7,62,359.237
七六二,三五九.二三七
```

isInteger() Method:

isInteger() Method Check করে যে Run করা মানটি একটি পূর্ণসংখ্যা কিনা। এই Method টি একটি বুলিয়ান মান (True Or False) প্রদান করে যা নির্দেশ করে যে প্রদত্ত মানটি একটি পূর্ণসংখ্যা কিনা।

Example:
```
let num = 12;
console.log(Number.isInteger(num));
console.log(Number.isInteger(12.12));
```
Output:
```
true
false
```

isFinite() Method:

isFinite() Method পরীক্ষা করে যে Run করা মানটি একটি Finite সংখ্যা কিনা। এই Method টি একটি Boolean মান (True Or False) প্রদান করে যা নির্দেশ করে যে প্রদত্ত মানটি Finite বা (সসীম) কি না।

Example:
```
let num1 = 386483265486;
console.log(Number.isFinite(num1));
let num3 = Infinity;
```

```
console.log(Number.isFinite(num3));
```
Output: true
false

isSafeInteger() Method:

isSafeInteger() Method একটি মান একটি নিরাপদ পূর্ণসংখ্যা কিনা তা পরীক্ষা করে। এই Method টি একটি Boolean মান (True Or False) প্রদান করে যা নির্দেশ করে যে প্রদত্ত মানটি একটি SafeInteger বা (নিরাপদ পূর্ণসংখ্যা) কিনা।

Example:
```
let num = 12;
console.log(Number.isSafeInteger(num));
let num1 = Infinity;
console.log(Number.isSafeInteger(num1));
```
Output: true
False

isNaN() Method:

Javascript -এ NaN হল "Not-a-Number" এর জন্য সংক্ষিপ্ত। একটি মান NaN হলে isNaN() Method টি True প্রদান করে। isNaN() Method টি পরীক্ষা করার আগে মানটিকে একটি সংখ্যায় রূপান্তর করে।

Example:
```
let num1 = NaN;
console.log(Number.isNaN(num1));
let num2 = "NaN";
console.log(Number.isNaN(num2));
let num3 = Infinity;
console.log(Number.isNaN(num3));
```
Output: true
false
false

*Javascript Math Methods

Javascript -এ Math হল একটি Built-in Object যা আপনাকে সংখ্যার ধরনে গাণিতিক ক্রিয়াকলাপ সম্পাদন করতে দেয়। Math একটি Constructor Function নয়। এটি Implicit Global অবজেক্টের একটি Property।

Math Methods in JavaScript are**:** Math **abs**(x), Math **sqrt**(x), Math **pow**(x, y), Math **max**(x, y, z, ..., n), Math **min**(x, y, z, ..., n), Math **cbrt**(x), Math **ceil**(x), Math **floor**(x), Math **round**(x), Math **random**(), Math **acos**(x), Math **acosh**(x), Math **asin**(x), Math **asinh**(x), Math **atan**(x), Math **atan2**(y, x), Math **atanh**(x), Math **cos**(x), Math **cosh**(x), Math **exp**(x), Math **log**(x), Math **sign**(x), Math **sin**(x), Math **sinh**(x), Math **tan**(x), Math **tanh**(x), Math **trunc**(x)

Math.abs(x):

abs() Method একটি সংখ্যার Absolute Value বা (পরম মান) প্রদান করে। নেগেটিভ বা পজেটিভ যে নাম্বারই ইনপুট দেওয়া হোক না কেন, পজেটিভ নাম্বারই রির্টান করবে।

Example:
```
let num1 = 32;
let num2 = -13;
let num3 = 4.76;
let num4 = 0;
console.log(Math.abs(num1));
console.log(Math.abs(num2));
console.log(Math.abs(num3));
console.log(Math.abs(num4));
```
Output:
```
32
13
4.76
0
```

Math.sqrt(x):

sqrt() Method একটি সংখ্যার বর্গমূল প্রদান করে।

Example:
```
let num1 = 144;
let num2 = -1;
let num3 = Infinity;
console.log(Math.sqrt(num1));
console.log(Math.sqrt(num2));
console.log(Math.sqrt(num3));
```
Output:
```
12
NaN
Infinity
```

Math.pow(x, y):

pow() Method টি দ্বিতীয় আর্গুমেন্টকে প্রথম আর্গুমেন্টের পাওয়ারে উন্নীত করে একটি সংখ্যার শক্তি গণনা করে। যেমনঃ $(4^2)=16$.

Example:
```
let power = Math.pow(4, 3);
  console.log(power);
  console.log(Math.pow(1, Infinity));
  console.log(Math.pow(-1, 2));
  console.log(Math.pow(-1, 3));
```
Output:
```
64
NaN
1
-1
```

Math.max(x, y, z, ..., n):

max() Method নির্দিষ্ট মানগুলির মধ্যে সর্বাধিক মান বা সর্বোচ্চ বড় মান খুঁজে বের করে এবং এটি প্রদান করে।

Example:
```
let digit = Math.max(12, 4, 5, 9, 0, -3);
  console.log(digit);
  console.log(Math.max(2947, -435, -123, 0, Infinity));
  console.log(Math.max(2947, -435, -123, 0, -Infinity));
```
Output:
```
12
Infinity
2947
```

Math.min(x, y, z, ..., n):

min() Method নির্দিষ্ট মানগুলির মধ্যে সর্বনিম্ন ছোট মান খুঁজে বের করে এবং এটি প্রদান করে।

Example:
```
let digit = Math.min(12, 4, 5, 9, 0, -3);
  console.log(digit);
  console.log(Math.min(2947, -435, -123, 0, Infinity));
  console.log(Math.min(2947, -435, -123, 0, -Infinity));
```
Output:
```
-3
-435
-Infinity
```

Math.cbrt(x):

cbrt() Method একটি নির্দিষ্ট সংখ্যার ঘনমূল গণনা করে এবং এটি প্রদান করে।

Example:
```
let cubeRoot = Math.cbrt(27);
```

```
console.log(cubeRoot);
console.log(Math.cbrt(125));
```

Output: 3
5

Math.ceil(x):

ceil() Method একটি দশমিক সংখ্যাকে পরবর্তী বৃহত্তম পূর্ণসংখ্যা পর্যন্ত Returns করে এবং এটি প্রদান করে। অর্থাৎ, 4.3 কে রাউন্ড করা হবে 5 (পরবর্তী বৃহত্তম পূর্ণসংখ্যা)।

Example:
```
let ceilNumber = Math.ceil(4.3);
console.log(ceilNumber);
console.log(Math.ceil(0.92));
```

Output: 5
1

Math.floor(x):

Math.floor(x) x এর মানকে তার নিকটতম পূর্ণসংখ্যাতে বৃত্তাকার করে: x এর থেকে কম বা সমান বৃহত্তম পূর্ণসংখ্যা প্রদান করে। অর্থাৎ, 4.3 কে রাউন্ড করা 4 হবে (নিকটতম পূর্ণসংখ্যা)।

Example:
```
let floorNumber = Math.ceil(4.3);
console.log(floorNumber);
console.log(Math.ceil(0.92));
```

Output: 4
0

Math.round(x):

Math.round() ফাংশন নিকটতম পূর্ণসংখ্যার বৃত্তাকার সংখ্যার মান প্রদান করে। Math.round(x) নিকটতম পূর্ণসংখ্যা প্রদান করে।

Example:
```
const num1 = 2.6;
console.log(Math.round(num1));
console.log(Math.round(2.5));
console.log(Math.round(2.4));
```

Output: 3
3
2

Math.random(x):

Math.random() 0 (Inclusive) এবং 1 (Exclusive) এর মধ্যে একটি Random (এলোমেলো) সংখ্যা প্রদান করে। 0 এবং 1 এর মধ্যে একটি Pseudo-Random সংখ্যা প্রদান করে। 0 এবং 1 এর মধ্যে এক এক সময় এক এক এলোমেলো বা রেন্ডম নাম্বার প্রদান করবে।

Example:
```
const num1 = 3;
console.log(Math.random(num1));
console.log(Math.random(3));
console.log(Math.random());
```
Output:
```
0.8727790555016821
0.14916377897257238
0.14262841537650206
```

Example:
```
const number = Math.random() * 5;
const rounded = Math.round(number);
console.log(rounded);
```
Output: 4

Example:
```
const number = Math.random() * 5;
const rounded = Math.floor(number);
console.log(rounded);
```
Output: 3

*Javascript String Methods and Some Properties

জাভাস্ক্রিপ্টে, স্ট্রিংগুলি অক্ষরগুলির একটি ক্রম উপস্থাপন করতে এবং কাজ করতে ব্যবহৃত হয়। একটি স্ট্রিং একটি Object এর পাশাপাশি Primitive ডেটা টাইপ প্রতিনিধিত্ব করতে পারে। জাভাস্ক্রিপ্ট স্বয়ংক্রিয়ভাবে Primitive স্ট্রিংগুলিকে স্ট্রিং অবজেক্টে রূপান্তর করে যাতে স্ট্রিং Method ব্যবহার করা এবং এমনকি Primitive স্ট্রিংগুলির জন্যও Properties অ্যাক্সেস করা সম্ভব হয়।

Remember: নিচের সব Methods এবং Properties এর মধ্যে **constructor**, **length**, **prototype**, এই গুলো হলো Properties। Properties লিখার ক্ষেত্রে Bracket () দিতে হয় না Methods এর মতো।

String Methods in JavaScript are**: length**, **indexOf()**, **lastIndexOf()**, **search()**, **slice()**, **substring()**, **substr()**, **replace()**, **toUpperCase()**, **toLowerCase()**, **trim()**, **concat()**, **includes()**, **split()**, **charAt()**, **charCodeAt()**, **constructor**,

prototype, **endsWith()**, **fromCharCode()**, **localeCompare()**, **match()**, **repeat()**, **startsWith()**, **toLocaleLowerCase()**, **toLocaleUpperCase()**, **toString()**, **trimEnd()**, **trimStart()**, **valueOf()**

Length:

Length Properties একটি স্ট্রিং এর দৈর্ঘ্য প্রদান করে। একটি খালি স্ট্রিংয়ের দৈর্ঘ্যের বৈশিষ্ট্য হল 0। সুতরাং, একটি String এর মধ্যে কয়টি Character রয়েছে তা নির্নয় করে।

Example:
```
let string = "Rajib Ar-Rahmaan";
let len = string.length;
console.log(len);
let stri = "Full Stack Web Development course with Jhankar Mahbub";
console.log(stri.length);
console.log("I love Allah".length);
```
Output:
```
16
53
12
```

indexOf():

indexOf() Method টি একটি স্ট্রিং-এ একটি মানের প্রথম উপস্থিতির অবস্থান প্রদান করে। মান পাওয়া না গেলে indexOf() Method টি -1 প্রদান করে। indexOf() Method টি Case Sensitive।

Example1:
```
let crushNames = ["Safa", "Shishir", "Tamannaah", "Prachi", "Anushka", "Jisoo"];
// let crush = crushNames.indexOf("Jisoo");
let anotherWay = crushNames[3];
console.log(anotherWay);
```
Output: Prachi

Example2:
```
let indexOf1 = "Full stack web development course with p-hero";
console.log(indexOf1.indexOf("course"));
let indexOf2 = "Full stack web development course with p-hero";
console.log(indexOf2.indexOf("Programming"));
let indexOf3 = "Full stack web development course with p-hero";
console.log(indexOf3.indexOf("a", 3));
let indexOf4 = "Full stack web development course with p-hero web";
console.log(indexOf4.indexOf("web", 12));
```
Output: 27

-1
7
46

lastIndexOf():

lastIndexOf() Method টি একটি স্ট্রিং-এ একটি নির্দিষ্ট মানের শেষ ঘটনার সূচক (অবস্থান) প্রদান করে। lastIndexOf() Method শেষ থেকে শুরু পর্যন্ত স্ট্রিং Search করে। lastIndexOf() Method টি শুরু থেকে Index প্রদান করে (Position 0)। যদি মান পাওয়া না যায় তবে lastIndexOf() Method টি -1 প্রদান করে। lastIndexOf() Method টি Case Sensitive।

Example: let lastIndexOf1 = "Full stack web development course course with p-hero";
console.log(lastIndexOf1.lastIndexOf("course"));
let lastIndexOf2 = "Full stack web development course with p-hero";
console.log(lastIndexOf2.lastIndexOf("Programming"));
let lastIndexOf3 = "Full stack web development course with p-hero a";
console.log(lastIndexOf3.lastIndexOf("a", 45));
let lastIndexOf4 = "Full stack web development course with p-hero web";
console.log(lastIndexOf4.lastIndexOf("web", 45));

Output: 34
-1
7
11

search():

search() Method একটি প্রদত্ত স্ট্রিং এবং একটি Regular Expression এর মধ্যে একটি মিলের জন্য Search করে। একটি String এর মধ্যে কোনও Expression Value এর Index Number খুঁজে বের করে।

Example: let searchMethod = "Rajib Ar-Rahmaan";
console.log(searchMethod.search("Rahmaan"));
var str="JavaScript is a scripting language. Scripting languages are often interpreted";
console.log(str.search("scripting"));

Output: 9
16

slice():

slice() Method একটি স্ট্রিং এর একটি অংশ বের করে। slice() Method টি একটি নতুন স্ট্রিং-এ Extracted Part ফেরত দেয়। slice() Method মূল স্ট্রিং পরিবর্তন করে না। শুরু এবং শেষ পরামিতি নিষ্কাশন করার জন্য স্ট্রিং এর অংশ নির্দিষ্ট করে। প্রথম Character index no = 0, দ্বিতীয় Character index no = 1, একটি Negative সংখ্যা স্ট্রিংয়ের End থেকে Select করা শুরু করে।

Example:
```
let num1 = "Rajib Ar-Rahmaan";
console.log(num1.slice(9));
let num2 = "Rajib Ar-Rahmaan";
console.log(num2.slice(-10));

var message = "Javascript is fun";
let sliceMethod = message.slice(0, 10);
console.log(sliceMethod);

let message = "Javascript is fun";
console.log(message.slice(-0, -4));
```
Output:
```
Rahmaan
Ar-Rahmaan
Javascript
Javascript is
```

substring():

substring() Method টি start এবং end সূচকগুলির মধ্যে স্ট্রিংয়ের একটি নির্দিষ্ট অংশ প্রদান করে। substring() is similar to slice().

Remember: The substring() method does not accept negative values.

Syntex: str.substring(indexStart, indexEnd)

Example:
```
let text = "Javascript is toy language";
console.log(text.substring(4, 10));
let text = "Javascript is toy language";
console.log(text.substring(-4));
```
Output:
```
script
Javascript is toy language
```

substr():

substr() Method একটি String এর একটি অংশ বের করে। substr() Method একটি নির্দিষ্ট অবস্থানে শুরু হয়

এবং একটি নির্দিষ্ট সংখ্যক অক্ষর প্রদান করে। substr() Method মূল String পরিবর্তন করে না। প্রথম Character index no = 0, দ্বিতীয় Character index no = 1, একটি Negative সংখ্যা স্ট্রিংয়ের End থেকে Select করা শুরু করে।

Syntex: string.substr(start,length)
Example: let text = "Hello World!";
console.log(text.substr(0, 5));
let text = "Hello World!";
console.log(text.substr(5));
let text = "Javatpoint";
console.log(text.substr(-5, 5));
Output: Hello
World!
Point

replace():

replace() Method একটি মান বা একটি Regular Expression এর জন্য একটি String Search করে। replace() Method Replace মান (গুলি) সহ একটি নতুন String প্রদান করে। replace() Method মূল String পরিবর্তন করে না।

Syntex: string.replace(originalstr,newstr)
Example: let text = "Microsoft is the best IT company in the world!"
console.log(text.replace("Microsoft", "Google"));
Output: Google is the best IT company in the world!

toUpperCase():

toUpperCase() Method একটি String -কে বড় হাতের অক্ষরে রূপান্তর করে।

Syntex: string.toUpperCase()
Example: let text = "Microsoft is the best IT company in the world!"
console.log(text.toUpperCase());
Output: MICROSOFT IS THE BEST IT COMPANY IN THE WORLD!

toLowerCase():

toLowerCase() Method একটি String -কে ছোট হাতের অক্ষরে রূপান্তর করে।

Syntex: string.toLowerCase()
Example: let text = "Microsoft is the best IT company in the world!"

```
console.log(text.toLowerCase());
```
Output: microsoft is the best it company in the world!

trim():

trim() Method একটি String এর উভয় প্রান্ত থেকে Whitespace অপসারণ করে।

Example:
```
let text = "   Rajib Ar-Rahmaan      ";
console.log(text.trim());
```
Output: Rajib Ar-Rahmaan

concat():

জাভাস্ক্রিপ্টে, আপনি দুই বা ততোধিক স্ট্রিং একসাথে সংযুক্ত করতে concat() পদ্ধতি ব্যবহার করতে পারেন। concat() পদ্ধতি একটি নতুন স্ট্রিং প্রদান করে যা concat() পদ্ধতিতে স্ট্রিংগুলির সংমিশ্রণ।

Example:
```
let text1 = "Rajib ";
let text2 = "Ar-";
let text3 = "Rahmaan";
console.log(text1.concat(text2, text3));
```
Output: Rajib Ar-Rahmaan

Example:
```
const string1 = "Hello";
const string2 = " beautiful";
const string3 = " World";
const newString = string1.concat(string2, string3);
console.log(newString);
```
Output: Hello beautiful World

includes():

জাভাস্ক্রিপ্টে Array.prototype.includes() পদ্ধতি একটি Boolean প্রদান করে যা নির্দেশ করে যে একটি অ্যারে এর উপাদানগুলির মধ্যে একটি নির্দিষ্ট মান অন্তর্ভুক্ত করে কিনা।

Syntex: string.includes(searchString[, position])

Example1:
```
let string = "Hello World!";
let searchString = "Hello";

let result = string.includes(searchString);
console.log(result); // true
```

Output: true

Example2:
```
let email = "example@gmail.com";
let search = "@gmail";
console.log(email.includes(search));
```
Output: true

*Javascript Array Methods

JavaScript-এ একটি Array হল এক ধরনের Global Object যা Data সঞ্চয় করতে ব্যবহৃত হয়। Array -গুলি একটি একক ভেরিয়েবলে একাধিক মান সংরক্ষণ করতে পারে, যা আমাদের Code -কে Condense এবং Organize করতে পারে।

Example:
```
const number = [12, 33, 45, 87, 98];
console.log(number);
```
Output: 12, 33, 45, 87, 98

Example:
```
const companyName = [
    "Google",
    "Meta",
    "Microsoft",
    "Tesla"
  ];
console.log(companyName);
```
Output: Google', 'Meta', 'Microsoft', 'Tesla'

Remember: নিচের সব Methods এবং Properties এর মধ্যে **constructor**, **length**, **prototype**, এই গুলো হলো Properties। Properties লিখার ক্ষেত্রে Bracket () দিতে হয় না Methods এর মতো।

Array Methods in JavaScript are**: push()**, **pop()**, **shift()**, **unshift()**, **splice()**, **includes()**, **isArray()**, **concat()**, **constructor**, **copyWithin()**, **entries()**, **every()**, **fill()**, **filter()**, **find()**, **findIndex()**, **forEach()**, **from()**, **indexOf()**, **join()**, **keys()**, **lastIndexOf()**, **length**, **map()**, **prototype**, **reduce()**, **reduceRight()**, **reverse()**, **slice()**, **some()**, **sort()**, **toString()**, **valueOf()**.

Remember: Array Traversing এর পরে Array Methods এর Definition নিচে দেওয়া হলো।

Array Traversing:

Javascript Array Elements/Objects অতিক্রম করার জন্য কোনো নির্দিষ্ট in-built function অফার করে না। আপনি loop ব্যবহার করে বা সরাসরি Element index দ্বারা একটি Array অতিক্রম করতে পারেন। একটি Array একই ধরনের একাধিক Element ধারণ করে, যা loop ব্যবহার করে **Traverse** করা যেতে পারে।

Example:
```
let numbers = [12, 13, 43, 55, 54, 33, 88];
for (let i = 0; i < numbers.length; i++) {
    console.log(numbers[i]);
}
```
Output:
```
12
13
43
55
54
33
88
```

Example:
```
let numbers = [12, 13, 43, 55, 54, 33, 88];
let sum = 0;
for (let i = 0; i < numbers.length; i++) {
    sum = sum+numbers[i];
}
console.log(sum);
```
Output:
```
298
```

Example:
```
let numbers = [12, 13, 43, 55, 54, 33, 88];
let sum = 0;
for (let i = 0; i < numbers.length; i++) {
    sum = sum+numbers[i];
    console.log(sum);
};
```
Output:
```
12
25
68
```

123
177
210
298

push():

push() Method টি অ্যারের শেষে এক বা একাধিক Item যোগ করে এবং অ্যারের নতুন Length প্রদান করে।

Syntex: array.push(element1, ..., elementN);

Example: let citys = ["Sylhet", "Dhaka", "Khulna"]
citys.push("Barisal");
console.log(citys);
Output: 'Sylhet', 'Dhaka', 'Khulna', 'Barisal'

Example: let frotsName = ["Apple", "Banana", "Orange"];
frotsName.push("Mangho", "Pineapple")
console.log(frotsName);
Output: 'Apple', 'Banana', 'Orange', 'Mangho', 'Pineapple'

pop():

পপ() Method একটি অ্যারে থেকে শেষ Item টি Remove করে দেয় এবং সেই উপাদানটি ফেরত দেয়।

Syntex: array.pop();

Example: let num = ["rajib", "sojib", "nabil", "sayem", "arif", "ibrahim"];
num.pop();
console.log(num);
Output: 'rajib', 'sojib', 'nabil', 'sayem', 'arif'

shift():

shift() Method একটি অ্যারের প্রথম Item Remove করে দেয়। shift() Method Original অ্যারে Change করে। shift() Method Shift করা Element -কে Return করে।

Syntex: array.shift();

Example: let num = ["rajib", "sojib", "nabil", "sayem", "arif", "ibrahim"];
num.shift();
console.log(num);
Output: 'sojib', 'nabil', 'sayem', 'arif', 'ibrahim'

unshift():

unshift() Method একটি অ্যারের শুরুতে এক বা একাধিক নতুন Iterm Add করে। unshift() Method Original অ্যারে ওভাররাইট করে।

Syntex: array.unshift(element1, ..., elementN);

Example1: let numbers = [12, 13];
numbers.unshift(10, 11);
console.log(numbers);
Output: 10, 11, 12, 13

Example2: let skills = ["Java","Python", "C",];
skills.unshift("Javascript", "PHP");
console.log(skills);
Output: 'Javascript', 'PHP', 'Java', 'Python', 'C'

splice():

Splice() Method অ্যারেতে Item Add করে And/Or Remove করে। splice() Method Original অ্যারে ওভাররাইট করে।

Syntex: array.splice(index, howMany, [element1][, ..., elementN]);

Example1: let skills = ["Javascript", "PHP", "Java", "Python", "C",];
skills.splice(1, 0, "C++");
console.log(skills);
Output: 'Javascript', 'C++', 'PHP', 'Java', 'Python', 'C'

Example2: let skills = ["Javascript", "PHP", "Java","Python", "C",];
skills.splice(1, 1, "C++");

```
console.log(skills);
```
Output: 'Javascript', 'C++', 'Java', 'Python', 'C'

Example3:
```
let skills = ["Javascript", "PHP", "Java","Python", "C",];
skills.splice(2, 2, "C++", "Rubby");
console.log(skills);
```

Output: 'Javascript', 'PHP', 'C++', 'Rubby', 'C'

Example4:
```
let num = ["rajib", "sojib", "nabil", "sayem", "arif", "ibrahim"];
num.splice(3, 1);
console.log(num);
```
Output: 'rajib', 'sojib', 'nabil', 'arif', 'ibrahim'

includes():

জাভাস্ক্রিপ্টে Array.prototype.includes() পদ্ধতি একটি Boolean প্রদান করে যা নির্দেশ করে যে একটি অ্যারে এর উপাদানগুলির মধ্যে একটি নির্দিষ্ট মান অন্তর্ভুক্ত করে কিনা।

Syntex: array.includes(valueToFind[, fromIndex]);

Example1:
```
const numbers = [1, 2, 3, 4];
console.log(numbers.includes(3));
console.log(numbers.includes(5));
```
Output: true
False

isArray():

JavaScript-এ, আপনি Array.isArray() পদ্ধতি ব্যবহার করে নির্ধারণ করতে পারেন যে একটি মান একটি অ্যারে কিনা। Array.isArray() পদ্ধতিটি সত্য প্রদান করে যদি আর্গুমেন্টটি একটি অ্যারে হয় এবং অন্যথায় মিথ্যা হয়।

Syntex: Array.isArray(arg)

Example1:
```
const array = [1, 2, 3];
console.log(Array.isArray(array));
```

```
const notArray = 'not an array';
console.log(Array.isArray(notArray));
```
Output: true
false

*Javascript Operators

অপারেটর হল মানগুলির মধ্যে প্রতীক যা যোগ, বিয়োগ, গুণ এবং আরও অনেক কিছুর মত বিভিন্ন ক্রিয়াকলাপকে অনুমতি দেয়। জাভাস্ক্রিপ্টের কয়েক ডজন অপারেটর রয়েছে।

There are different types of JavaScript Operators**: Arithmetic** Operators, **Comparison** Operators, **Logical** Operators, **Assignment** Operators, **Conditional** Operators, **Type** Operators.

Arithmetic Operators: Arithmetic Operators সাংখ্যিক Oparend গুলির মধ্যে Arithmetic ক্রিয়াকলাপ সম্পাদন করতে ব্যবহৃত হয়।

Arithmetic Operators are**: +** (Addition), **-** (Subtraction), ***** (Multiplication), */* (Division), ****** Exponentiation (ES2016), **%** Modulus (Remainder), **++** (Increment), **–** (Decrement).

Example (Addition):
```
let  num1 = 12;
let num2 = 2;
let output = num1 + num2;
console.log(output);
```
Output: 14
Example (Subtraction):
```
let  num1 = 12;
let num2 = 2;
let output = num1 - num2;
console.log(output);
```
Output: 10
Example (Multiplication):
```
let  num1 = 12;
let num2 = 2;
let output = num1 * num2;
```

```
console.log(output);
```

Output: 24

Example (Division):
```
let  num1 = 12;
let num2 = 2;
let output = num1 / num2;
console.log(output);
```

Output: 6

Example (Exponentiation):
```
let  num1 = 12;
let num2 = 2;
let output = num1 ** num2;   // 12 ^ 2 = 144
console.log(output);
```

Output: 144

Example (Modulus/Remainder):
```
let  num1 = 11;
let num2 = 2;
let output = num1 % num2;   // [ ভাগশেষ বের করা হয়। ]
console.log(output);
```

Output: 1

Example (Increment):
```
let i = 12;
i++       // i = i + 1;
console.log(i);
```

Output: 13

Example (Decrement):
```
let i = 12;
i--       // i = i - 1;
console.log(i);
```

Output: 11

Comparison Operators: Javascript Comparison Operators প্রদান করে যে দুটি Oparend এর মধ্যে তুলনা করে এবং একটি Boolean মান true Or false প্রদান করে।

Comparison Operators are**: >** (greater than), < (less than), **==** (equal to), **===** (equal value and equal type), **!=** (not equal), **!==** (not equal value or not equal type), **>=** (greater than or equal to), **<=** (less than or equal to).

Example (Greater than):
```
let a = 12;
let b = 4;
```

```
if (a > b) {
    console.log("A is Greater than B");
}   else {
    console.log("A is not Greater than B");
}
```

Output: A is Greater than B

Example (Less than):

```
let a = 12;
let b = 4;

if (a < b) {
    console.log("A is Greater than B");
} else {
    console.log("A is not Greater than B");
}
```

Output: A is not Greater than B

Example (equal to):

```
let a = "12";
let b = 10;
                    // Equal to inner value
if (a == b) {
    console.log("The statement is  true");
} else {
    console.log("The statement is false");
}
```

Output: The statement is false

Example (equal value and equal type):

```
let a = "12";
let b = 12;
        // Equal value and equal type
if (a === b) {
    console.log("This statement is true")
} else {
    console.log("This statement is false");
}
```

Output: This statement is false

Example (Not equal):

```
let a = "12";
let b = 12;
```

```
                    // Not equal to inner value
if (a != b) {
    console.log("This statement is true")
} else {
    console.log("This statement is false");
}
```

Output: This statement is false

Example (Not equal value or not equal type):

```
let a = "12";
let b = 12;
        // Not equal value or not equal type
if (a !== b) {
    console.log("This statement is true")
} else {
    console.log("This statement is false");
}
```

Output: This statement is true

Example (Greater than or equal to):

```
let a = 12;
let b = 12;
        // Greater than or equal to
if (a >= b) {
    console.log("This statement is true")
} else {
    console.log("This statement is false");
}
```

Output: This statement is true

Example (Less than or equal to):

```
let a = 13;
let b = 12;
        // Less than or equal to
if (a <= b) {
    console.log("This statement is true")
} else {
    console.log("This statement is false");
}
```

Output: This statement is false

Logical Operators: Javascript -এ তিনটি Logical Operators রয়েছে, যা একটি true ("truthy") Or false ("falsey") মান ফেরাতে দুই বা ততোধিক Programming Statement -কে সংযুক্ত করে। এগুলি প্রায়শই বুলিয়ান (Logical) প্রকারের সাথে ব্যবহৃত হয় তবে যে কোনও Data প্রকারের মানগুলিতে প্রয়োগ করা যেতে পারে।

Logical Operators are**:** **&&** (And), **||** (Or), **!** (Not).

Logical AND Operator: && উভয় Operand true হলে true মূল্যায়ন করে, অন্যথায় false -কে মূল্যায়ন করে।

Example:
```
let a = 3;
let b = -2;

console.log(a > 0 && b > 0);
```
Output: false

Example:
```
let a = 12;
let b = 10;

let myCondition = a > b && a != b;
console.log(myCondition);
```
Output: true

Logical Or Operator: || (Or) Oparend এর যেকোনো একটি true হলে true মূল্যায়ন করে। উভয় Oparend false হলে, ফলাফল false.

Example:
```
let a = 12;
let b = 10;

let myCondition = a > b || a == b;
console.log(myCondition);
```
Output: true

Example:
```
let a = 12;
let b = 12;

let myCondition = a > b || a != b;
console.log(myCondition);
```

Output: false

Logical Not Operator: ! (Not) Oparend false এবং বিপরীত হলে true মূল্যায়ন করে।

Example: let a = 12;
let b = 10;

let myCondition = !(a < b); // Not False = True
console.log(myCondition);

Output: true

Example: let a = 12;
let b = 10;

let myCondition = !(a != b); // Not True = False
console.log(myCondition);

Output: false

Assignment Operators: Assignment Operator গুলো Javascript Variable -এর মান নির্ধারণ করে।

Assignment Operators are**: =** (Simple Assignment Operator), **+=** (Addition assignment), **-=** (Subtraction Assignment), ***=** (Multiplication Assignment), **/=** (Division Assignment), **%=** (Remainder/Modulus Assignment), ****=**, (Exponentiation Assignment).

Simple Assignment Operator (=): Simple Assignment Operator একটি ভেরিয়েবলের জন্য একটি মান নির্ধারণ করে।

Example: let x = 10;
let y = 50 - x;
console.log(y);

Output: 40

Addition Assignment (+=): Addition Assignment Operator একটি ভেরিয়েবলে একটি মান যোগ করে।

Example: let x = 10;
x += 12;

```
console.log(x);
```
Output: 22

Example:
```
let x = "Rajib";
x += " Rahman";
console.log(x);
```
Output: Rajib Rahman

Subtraction Assignment (-=): Subtraction Assignment Operator একটি ভেরিয়েবলে একটি মান বিয়োগ করে।

Example:
```
let x = 22;
x -= 10;
console.log(x);
```
Output: 12

Multiplication Assignment (*=): Multiplication Assignment Operator একটি ভেরিয়েবলকে গুণ করে।

Example:
```
let x = 6;
x *= 2;
console.log(x);
```
Output: 12

Division Assignment (/=): Division Assignment Operator একটি ভেরিয়েবলকে ভাগ করে।

Example:
```
let x = 24;
x /= 2;
console.log(x);
```
Output: 12

Remainder/Modulus Assignment (%=): Reminder/Modulus Assignment Operator একটি ভেরিয়েবলের ভাগশেষ একটি অবশিষ্ট করে।

Example:
```
let x = 25;
x %= 2;
console.log(x);
```
Output: 1

Exponentiation Assignment (=):** Exponentiation Assignment Operator

অপারেন্ডের শক্তিতে একটি পরিবর্তনশীল উত্থাপন করে। যেমনঃ 10 ^ 2 = 100

Example: let x = 5;
x **= 2;
console.log(x);
Output: 25

Conditional (Ternary) Operators: Ternary Operator হল একটি Simplified Conditional Operator যেমন if/else।

Syntax: variable = (condition) ? value1: value2

Example: let number = 12 < 5 ? true : false;
console.log(number);
Output: false

Example: let age = 18;
let message;

if (age >= 16) {
  message = 'You can drive.';
} else {
  message = "You can't drive";
}

console.log(message);
Output: You can drive.

*Typeof Operator

জাভাস্ক্রিপ্টে **Typeof** হল একটি অপারেটর যা টাইপ চেকিংয়ের জন্য ব্যবহৃত হয় এবং এটিতে পাস করা অপারেন্ডের ডেটা টাইপ ফেরত দেয়। অপারেন্ড যেকোন ভেরিয়েবল, ফাংশন বা অবজেক্ট হতে পারে যার ধরন আপনি typeof অপারেটর ব্যবহার করে খুঁজে পেতে পারেন।

Example: let a = 10;
let b = true; // true =1, and false =0

```
let c = a + b;
console.log(typeof c);
```

Output: number

Example:
```
typeof "John"                 // Returns "string"
typeof 3.14                   // Returns "number"
typeof NaN                    // Returns "number"
typeof false                  // Returns "boolean"
typeof [1,2,3,4]              // Returns "object"
typeof {name:'John', age:34}  // Returns "object"
typeof new Date()             // Returns "object"
typeof function () {}         // Returns "function"
typeof myCar                  // Returns "undefined" *
typeof null                   // Returns "object"
```

*Javascript Controls Structures

Control Structures আসলে একটি প্রোগ্রামের Execution প্রবাহ নিয়ন্ত্রণ করে।

Javascript has **TWO** Primary types of **Control Structures: Conditional Statements** and **Controls Loops**.

Javascript Conditional Statements: Conditional Statements জাভাস্ক্রিপ্টে behavior controls করে এবং কোডের টুকরো চলতে পারে কিনা তা নির্ধারণ করে। জাভাস্ক্রিপ্টে বিভিন্ন ধরণের Conditionals রয়েছে যার মধ্যে রয়েছে: **"if"** Statement**:** যেখানে একটি Condition true হলে এটি কোডের একটি ব্লকের জন্য এক্সিকিউশন নির্দিষ্ট করতে ব্যবহৃত হয়।

Javascript **Conditional Statements** are**: if** statement, **else** statement, **else if** statement, **switch** statement.

If Statement: Conditional এর সবচেয়ে Common ধরন হিসাবে, **if** Statement টি তখনই চলে যদি বন্ধনী () এ থাকা Condition টি true হয়।

Example1:
```
if (10 > 5) {
    var outcome = "if block";
}
```

```
console.log(outcome);
```
Output: if block

Example2:
```
let age = 18;
let message;

if (age >= 16) {
  message = 'You can drive.';
}
console.log(message);
```
Output: You can drive.

Else Statement: If এর Condition টি False হলে Execute করার জন্য কোডের একটি Block নির্দিষ্ট করতে **else** Statement টি ব্যবহার করা হয়।

Example1:
```
let age = 18;
let message;

if (age == 16) {
    message = "You can drive.";
} else {
    message = "You can't drive.";
}
console.log(message);
```
Output: You can’t drive.

Example2:
```
let num = "rajib";
if (num != "rajib") {
  console.log("This is my name");
} else {
  console.log("This is not my name");
}
```
Output: This is not my name

Else If Statement: First Condition টি False হলে একটি New Condition নির্দিষ্ট করতে **else if** Statement টি ব্যবহার করা হয়।

Example1:
```
let num = "Rajib";
```

```
if (num == "Suhana") {
   console.log("Your name is " + num);
} else if (num == "Samiya") {
   console.log("Your name is " + num);
} else if (num == "Rajib") {
   console.log("Your name is " + num);
}
```

Output: Your name is Rajib

Example2: let time = 24;

```
if (time < 10) {
 greeting = "Good morning";
} else if (time < 20) {
 greeting = "Good day";
} else {
 greeting = "Good evening";
}
console.log(greeting);
```

Output: Good evening

Example3: let num = "Mohammad";

```
if (num == "Suhana") {
 console.log("Your name is " + num);
} else if (num == "Samiya") {
 console.log("Your name is " + num);
} else if (num == "Rajib") {
 console.log("Your name is " + num);
} else {
 console.log("Your name is " + num);
}
```

Output: Your name is Mohammad

Switch Statement: Switch Statements বিভিন্ন অবস্থার উপর ভিত্তি করে বিভিন্ন ক্রিয়া সম্পাদন করতে ব্যবহৃত হয়। Switch Statements ব্যবহার করে অনেকগুলো কোড ব্লকের মধ্যে একটি নির্বাচন করতে হবে।

Example1: let num = "Suhan";

```
switch (num) {
  case "Rajib":
    console.log("Your name is " + num);
    break;
  case "Suhan":
    console.log("Your name is " + num);
    break;
  case "Mohammad":
    console.log("Your name is " + num);
    break;
  default:
    console.log("Your name is " + num);
}
```

Output: Your name is Suhan

Example2:

```
const expr = 'Papayas';
switch (expr) {
  case 'Oranges':
    console.log('Oranges are $0.59 a pound.');
    break;
  case 'Mangoes':
  case 'Bananas':
    console.log('Mangoes and papayas are $2.79 a pound.');
    break;
  default:
    console.log(`Sorry, we are out of ${expr}.`);
}
```

Output: Sorry, we are out of Papayas.

Example3:

```
const country = "Bangladesh";
switch (country) {
  case "France":
  case "Spain":
  case "Ireland":
  case "Poland":
```

```
    console.log("This country is in Europe.");
    break;
  default:
    console.log("This country is not in Europe.");
}
```

Output: This country is not in Europe.

Example4:
```
let day = new Date().getDay();
let dayName;

switch (day) {
  case 0: dayName = "Sunday";
    break;
  case 1: dayName = "Monday";
    break;
  case 2: dayName = "Tuesday";
    break;
  case 3: dayName = "Wednesday";
    break;
  case 4: dayName = "Thursday";
    break;
  case 5: dayName = "Friday";
    break;
  case 6: dayName = "Saturday";
    break;
}
console.log("Today " + dayName);
```

Output: Today Tuesday

Javascript Controls Loops: কোডের একই block বারবার execute করতে Loop ব্যবহার করা হয়, যতক্ষণ না একটি নির্দিষ্ট Condition পূরণ হয়। একটি লুপের পিছনে মূল ধারণা হল সময় এবং প্রচেষ্টা বাঁচাতে একটি প্রোগ্রামের মধ্যে পুনরাবৃত্তিমূলক কাজগুলি স্বয়ংক্রিয় করা।

Javascript **Controls Loops** are**: For** Loops, **While** Loops, **Do While** Loops, **Nested** loops, **For Of** Loops and **For In** Loops.

For Loops: লুপের জন্য Javascript নির্দিষ্ট সময়ের জন্য Element গুলিকে Loop করে। পুনরাবৃত্তির সংখ্যা জানা থাকলে এটি ব্যবহার করা উচিত।

Syntax: for (initialization; condition; finalExpression) {
// code
}

Example1: for (let i = 0; i <= 5; i++) {
console.log(i);
}

Output: 1
2
3
4
5

Example2: let nums = ["samia", "rajib", "sojib", "sonia", "tania"];

for (let i = 0; i < nums.length; i++) {
console.log(nums[i]);
}

Output: samia
rajib
sojib
sonia
tania

Example3: for (let i = 1; i <= 10; i++) {
console.log(i);
if (i >= 5) {
break;
}
}

Output: 1
2
3
4
5

While Loops: Javascript যখন Loop Infinite সংখ্যার জন্য উপাদানগুলিকে পুনরাবৃত্তি করে। পুনরাবৃত্তির সংখ্যা জানা না থাকলে এটি ব্যবহার করা উচিত।

Syntax:
```
while(condition) {
    // Code to be executed
}
```

Example1:
```
let i = 1;

while (i < 5) {
  console.log(i);
  i++;
}
```

Output:
```
1
2
3
4
```

Example2:
```
let r = 2;
while ( r <= 10) {
  console.log(r);
  r += 2;
}
```

Output:
```
2
4
6
8
10
```

Example2:
```
let i = 1;
while (i <= 10) {
  console.log(i);
  i++;
  if (i > 5) {
    break;
  }
}
```

Output: 1
2
3
4
5

Do While Loops: do-while Loop হল while লুপের একটি বৈকল্পিক, JavaScript do while লুপ Infinite সংখ্যক বার এলিমেন্টের পুনরাবৃত্তি করে যেমন while লুপ। কিন্তু, Condition true Or false কিনা তা অন্তত একবার কোড কার্যকর করা হয়।

Syntax:
```
do {
    // Code to be executed
}
while(condition);
```

Example1:
```
let i = 1;
do {
  console.log(i);
  i++;
} while (i < 6);
```

Output: 1
2
3
4
5

Example2:
```
const myArray = [];
let i = 5;

do {
  myArray.push(i);
  i++;
} while (i < 5);

console.log(myArray);
```

Output: [5]

Nested Loops: Nested Loop হল একটি Loop যা অন্য লুপের ভিতরে থাকে। জাভাস্ক্রিপ্ট জাভাস্ক্রিপ্টে Nested Loop Supports করে। Loop টি তে এক বা একাধিক বা সাধারণ অন্য লুপের ভিতরে সংজ্ঞায়িত যেকোন সংখ্যক Loop থাকতে পারে এবং লুপের ভিতরে নেস্টিংয়ের n স্তরও আচরণ করতে পারে।

Syntax:
```
// Outer for loop.
for ( initialization; test-condition; increment/decrement )
{
// Inner for loop.
for ( initialization; test-condition; increment/decrement )
{
// statement of inner loop
}
// statement of outer loop
}
```

Example1:
```
let arr = [
    ["rajib", "sojib", "samia"],
    ["arif", "ibrahim", "himel"],
    ["talha", "coyon", "akram"],
];

for (let r = 0; r < arr.length; r++) {
  for (let s = 0; s < arr[r].length; s++) {
    console.log(arr[r][s]);
  }
}
```

Output:
```
rajib
sojib
samia
arif
ibrahim
himel
talha
coyon
Akram
```

For Of Loops: for-of হল ES6-এ একটি নতুন Loop যা for-in এবং forEach() উভয়কেই প্রতিস্থাপন করে এবং নতুন পুনরাবৃত্তি প্রোটোকল সমর্থন করে। পুনরাবৃত্তিযোগ্য Object (Arrays, Strings, Maps, Sets, etc) লুপ করতে এটি ব্যবহার করে।

Syntax:
```
for (variable of object) {
// code
}
```

Example1:
```
let arr = [1, 2, 3, 4];
for (let el of arr) {
  console.log(el);
}
```

Output:
```
1
2
3
4
```

Example2:
```
let num = "Rajibur Rahman";
for (let str of num) {
  console.log(str);
}
```

Output:
```
R
a
j
i
b
u
r

R
a
h
m
a
N
```

For In Loops: Statement –এ **for...** একটি Object এর বৈশিষ্ট্যের উপর কম্বো Return করে (Loop)। লুপের ভিতরে Code Block প্রতিটি Property জন্য একবার Execute করা হয়।

Syntax: for (x in object) {
code block to be executed
}

Example1:
```
const person = { fname: "John", lname: "Doe", age: 25 };
let text = "";
for (let x in person) {
  text += person[x] + " ";
}
console.log(text);
```

Output: John Doe 25

Example2:
```
let totn_colors = {
  primary: "blue",
  secondary: "gray",
  tertiary: "white",
};

for (let color in totn_colors) {
  console.log(totn_colors[color]);
}
```

Output: blue
gray
white

*Javascript Functions

একটি Javascript Function একটি নির্দিষ্ট কাজ সম্পাদন করার জন্য Design করা কোডের একটি Block। একটি Javascript Function Execute হয় যখন "Something" এটিকে Invoke করে (Call করে)। Javascript Function Operation সঞ্চালনে ব্যবহার করা হয়। কোড Reuse এর জন্য আমরা Javascript ফাংশনকে অনেকবার Call করতে পারি।

Syntex:
```
function name(parameter1, parameter2, parameter3) {
    // code to be executed
}
```

Example:
```
function greet() {
    console.log("Hello there");
```

```
}
greet();
```
Output: Hello there

Function Declaration: Function Keyword টি প্রথমে, তারপর Function name, তারপর বন্ধনীর মধ্যে Parameter গুলির একটি তালিকা (Comma দ্বারা পৃথক করা) এবং অবশেষে ফাংশনের Code, নামও দেওয়া হয়েছে " the function body", Curly brackets এর মধ্যে।

Syntex:
```
function name(param0) {
  statements
}
function name(param0, param1) {
  statements
}
function name(param0, param1, /* … ,*/ paramN) {
  statements
}
```

Example:
```
function calcRectArea(width, height) {
   return width * height;
  }

  console.log(calcRectArea(5, 6));
```
Output: 30

Function Invoke (Calling): Javascript ফাংশনের ভিতরের কোডটি কার্যকর হবে যখন "Something" এটিকে Call করে।

Syntex:
```
function myFunction( var ) {
    return var;
}
myFunction( value );
```

Example:
```
function myFunction(a, b) {
  console.log(a * b);
}
myFunction(10, 2);
```

Output: 20

Function Parameter: Function Parameter -গুলি হল সেই নামগুলি যা ফাংশনের সংজ্ঞায় সংজ্ঞায়িত করা হয় এবং ফাংশনের সংজ্ঞায় ফাংশনে পাস করা আসল মানগুলি Arguments হিসাবে পরিচিত।

Syntex:
```
function Name(parameter1, parameter2, parameter3, parameter4) {
    // Statements
}
```

Example:
```
function example(parameter) {
  console.log(parameter); // Output = foo
}

const argument = "foo";

example(argument);
```
Output: foo

Function Arguments: Function কলে Parameter -গুলি হল ফাংশনের Argument। Javascript Argument -গুলি মান দ্বারা পাস করা হয়: ফাংশনটি শুধুমাত্র মানগুলি জানতে পারে, আর্গুমেন্টের অবস্থানগুলি নয়। যদি একটি ফাংশন একটি আর্গুমেন্টের মান পরিবর্তন করে তবে এটি প্যারামিটারের মূল মান পরিবর্তন করে না।

Example:
```
function func1(a, b, c) {
  console.log(arguments[0]);
  console.log(arguments[1]);
  console.log(arguments[2]);
}

func1("Rajib", "Suhana", "Tanhar");
```
Output:
```
Rajib
Suhana
Tanhar
```

Function Return: যখন Javascript একটি Return স্টেটমেন্টে পৌঁছায়, Function -টি Execute করা বন্ধ করবে। যদি একটি Statement থেকে Function -টি চালু করা হয়, তাহলে Javascript Invoking

স্টেটমেন্টের পরে কোডটি Execute করতে "Return" করবে। ফাংশন প্রায়ই একটি Return মান গণনা করে। ফেরত মান "কলার": -এ ফেরত দেওয়া হয়।

Syntex: return value;

Example1:
```
function addition(x, y) {
  return (x + y);
}
let store = addition(12, 12);
console.log(store);
```
Output: 24

Example2:
```
function addition(x, y) {
  let A = x * y;
  return A;
}
let store = addition(12, 12);
console.log(store);
```
Output: 144

Function Expression: জাভাস্ক্রিপ্টে ফাংশন তৈরি করার দুটি ভিন্ন উপায় রয়েছে। আমরা Function Declaration এবং Function Expression ব্যবহার করতে পারি। তাদের মধ্যে পার্থক্য হল Function name, যা Anonymous(বেনামী) Function তৈরি করতে Function এক্সপ্রেশনে বাদ দেওয়া যেতে পারে।

Syntex:
```
let welcome = function () {
  console.log("Welcome to W3Docs!");
};
welcome();
```

Example1:
```
const square = function (number) {
    return number * number;
};
const x = square(4);
console.log(x);
```
Output: 16

Example2:
```
// program to find the square of a number
// function is declared inside the variable
```

```
let x = function (num) {
  return num * num;
};
console.log(x(4));

// can be used as variable value for other variables
let y = x(3);
console.log(y);
```

Output: 16
9

Function Higher Order: Basically, একটি Function যা অন্য একটি Function -কে Argument হিসেবে নেয় বা একটি Function Return করে তাকে Higher Order Function বলে। অর্থাৎ: একটি Function -কে অন্য ফাংশনে Argument হিসেবে পাস করা। অন্য Function থেকে একটি Function Return করা।

Example1:

```
const numbers = [1, 2, 3, 4, 5];

function addOne(array) {
  for (let i = 0; i < array.length; i++) {
    console.log(array[i] + 1);
  }
}

addOne(numbers);
```

Output: 2
3
4
5
6

Function Callback: একটি Callback Function হল একটি Function যা একটি Argument হিসাবে অন্য ফাংশনে পাস করা হয়, যেটিকে বাইরের ফাংশনের ভিতরে কিছু ধরণের Rutin Or Action Complete করার জন্য Call করা হয়।

Example:

```
const add = function (a, b) {
```

```
  return a + b;
};
const minus = function (a, b) {
  return a - b;
};
const multiply = function (a, b) {
  return a * b;
};
const division = function (a, b) {
  return a / b;
};
function cals(num1, num2, callBack) {
  return callBack(num1, num2);
}
console.log(cals(12, 10, minus));
```

Output: 2

*Javascript Objects

Object Type জাভাস্ক্রিপ্টের Data টাইপগুলির একটিকে Represent করে। এটি বিভিন্ন Keyed Collection এবং আরও Complex Entities Store করতে ব্যবহৃত হয়। Object() Constructor Or Object Initializer/Literal Syntax ব্যবহার করে Object তৈরি করা যায়।

একটি Javascript Object হল একটি Entity যার State এবং Behavior (Properties And Methods)। যেমন: Car, Pen, Bike, Chair, Glass, Keyboard, Monitor etc। Javascript একটি Object-based Language। সবকিছুই জাভাস্ক্রিপ্টের একটি Object।

Syntex:
```
let object_name = {
    key_name : value,
    ....
}
```

Example:
```
let school = {
    name: "Vivekananda School",
    location: "Delhi",
    established: "1971",
    displayInfo: function () {
      console.log(`${school.name} was established
```

```
                    in ${school.established} at ${school.location}`);
    },
};
school.displayInfo();
```

Output: Vivekananda School was established
in 1971 at Delhi

Object Properties: একটি Javascript Object Basically a collection of unordered Properties এর একটি Collection। Javascript অবজেক্টের সাথে যুক্ত Value -কে এর Properties বলা হয়। শুধুমাত্র পঠনযোগ্য Properties বাদ দিয়ে Properties সাধারণত যোগ করা, আপডেট করা এবং মুছে ফেলা যায়।

Syntax1: objectName.property // person.age
Syntax2: objectName["property"] // person["age"]
Syntax3: objectName[expression] // x = "age"; person[x]

Example1:
```
let user = {
  // an object
  name: "Rajib", // by key "name" store value "Rajib"
  age: 30, // by key "age" store value 30
};
console.log(user.name);
```

Output: Rajib

Example2:
```
let person = {
  firstname: "Rajib",
  lastname: "Rahman",
  age: 21,
  eyecolor: "slightly red",
};
console.log(
   person["firstname"] +
      " " +
      person["lastname"] +
      " is " +
      person["age"] +
      " years old."
);
```

Output: Rajib Rahman is 21 years old.

Object Methods: জাভাস্ক্রিপ্টের Object হল Key/Value জোড়ার Collections। মানগুলি Properties এবং Methods নিয়ে গঠিত হতে পারে এবং অন্যান্য সমস্ত জাভাস্ক্রিপ্ট Data প্রকার থাকতে পারে, যেমন String, Number and Boolean। জাভাস্ক্রিপ্টের সমস্ত Object parent Object Constructor থেকে আসে।

The various **Methods of Object** are as follows**:** Object.**keys()**, Object.**values()** Object.**assign()**, Object.**create()**, Object.**defineProperty()**, Object.**defineProperties()**, Object.**entries()**, Object.**freeze(),** Object.**getOwnPropertyDescriptor()**, Object.**getOwnPropertyDescriptors()**, Object.**getOwnPropertyNames()**, Object.**getOwnPropertySymbols()**, Object.**getPrototypeOf()**, Object.**is()**, Object.**isExtensible()**, Object.**isFrozen()**, Object.**isSealed()**, Object.**preventExtensions()**, Object.**seal()**, Object.**setPrototypeOf()**,
Syntax: objectName.methodName()

Example1:
```
const person = {
  firstName: "John",
  lastName: "Doe",
  id: 5566,
  fullName: function () {
    return this.firstName + " " + this.lastName;
  },
};
console.log(person.fullName());
```
Output: John Doe
Example2:
```
let user = {
  name: "John",
  age: 30
};
user.sayHi = function() {
  alert("Hello!");
};
user.sayHi(); // Hello!
```

Object.keys(): Object.keys() হল একটি জাভাস্ক্রিপ্ট পদ্ধতি যা একটি প্রদত্ত বস্তুর নিজস্ব গণনাযোগ্য প্রপার্টির নামগুলির একটি অ্যারে প্রদান করে, যে ক্রমে আমরা একটি সাধারণ লুপের সাহায্যে পাই।

Syntax: Object.keys(object)

Example1:
```
const obj = { name: "John", age: 30, city: "New York" };
const keys = Object.keys(obj);
console.log(keys);
```
Output: ["name", "age", "city"]

Object.values(): Object.values() হল জাভাস্ক্রিপ্টের একটি পদ্ধতি যা একটি প্রদত্ত অবজেক্টের নিজস্ব গণনাযোগ্য সম্পত্তির মানগুলির একটি অ্যারে প্রদান করে, যেমনটি আমরা একটি সাধারণ লুপের সাথে পাই।

Syntax: Object.keys(object)

Example1:
```
const obj = { name: "John", age: 30, city: "New York" };
const values = Object.values(obj);
console.log(values);
```
Output: ["John", 30, "New York"]

Object Literal: জাভাস্ক্রিপ্টে Literal একটি Object আমাদেরকে Plain Javascript Object তৈরি করতে দেয়। এতে Key-Value এর জোড়ার একটি List থাকে, প্রতিটি একটি Comma দ্বারা পৃথক করা হয় এবং Curly Brackets এর ভিতরে Wrapp হয়।

Syntax1:
```
var person = {
    firstName: "Allie",      //FirstName Property
    lastName: "Grater",      //lastName Property
    age: 50                  //age
};
```
Syntax2: object={property1:value1,property2:value2.....propertyN:valueN}

Example1:
```
let emp = {
  id: 102,
  name: "Shyam Kumar",
  salary: 40000,
};
console.log(emp);
```
Output: {id: 102, name: 'Shyam Kumar', salary: 40000}

Example2:
```
let anotherPerson = {
  name: "Rajib Rahman",
  Address: "Sunamganj",
  Age: 21,
  Nationality: "Bangladeshi",
  birthYear: function () {
    return 2022 - 21;
  },
};
console.log(anotherPerson.birthYear());
```
Output: 2001

Object Delete Operator: Delete Operator একটি Object থেকে একটি Property Delete দেয়। Delete দেওয়ার Operation টি Original Object কে পরিবর্তন করে। এর মানে হল এটি একটি পরিবর্তনযোগ্য অপারেশন।

Syntax1:
```
delete object.property
delete object[property]
```

Example1:
```
let person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue",
};
delete person.age;
console.log(person);
```
Output: { firstName: 'John', lastName: 'Doe', eyeColor: 'blue' }

Example2:
```
const Employee = {
  firstname: "Rajib",
  lastname: "Doe",
};

console.log(Employee.firstname);
// expected output: "John"

delete Employee.firstname;
```

```
    console.log(Employee.firstname);
    // expected output: undefined
```

Output: Rajib
Undefined

*Javascript Recursion

Recursion হল যখন একটি ফাংশন নিজেই কল করে যতক্ষণ না কেউ এটিকে থামায়। এটি একটি লুপের পরিবর্তে ব্যবহার করা যেতে পারে। যদি কেউ এটি বন্ধ না করে তবে এটি চিরতরে পুনরাবৃত্তি হবে এবং আপনার প্রোগ্রামটি ক্র্যাশ করবে। একটি বেস কেস একটি শর্ত যা পুনরাবৃত্তি বন্ধ করে।

Syntax1:
```
function recurse() {
    // function code
    recurse();
    // function code
}

recurse();
```

Example:
```
function log(num){
    if(num > 5){
      return;
    }
    console.log(num);
    log(num + 1);
}

log(1);
```

Output: 1
2
3
4
5