ETHEREUM 2.0 MASTERY PROGRAM

Instructor: Raja Rizwan Saleem

Module

ONE

JAVASCRIPT CRASH COURSE

Instructor: Raja Rizwan Saleem

JAVASCRIPT VARIBLES

WHAT VARIABLE IS IN JS?

In any programming language, we typically do lots of calculations. The calculation results are stored in the computer's memory. Just like human memory, the memory of the computer also consists of millions of cells. The calculated values are stored in these memory cells. To make the usage and retrieval of these values easy, these memory locations are given names. The names given to these locations are called variables.

WHAT VARIABLE IS IN JS?

- Variable means anything that can vary.
- JavaScript includes variables which hold the data value and it can be changed anytime.
- Variable is the name of the storage location.
- When we want to save some data, we store it in a variable
- JavaScript uses reserved keyword **var** to declare a variable. A variable must have a unique name. You can assign a value to a variable using equal to **(=)** operator when you declare it or before using it.

WHAT VARIABLE IS IN JS?

When the variable is declared, the JavaScript engine assigns it a memory or space. Because of this, once a variable is declared, it takes a value of undefined even before the assignment. We assigned data to the variable by using the assignment operator "=". Datatypes in JavaScript are:

- Number i.e., 11,23,45,6
- Strings, i.e., "Hello World."
- Boolean, i.e., true, false
- Undefined
- Null
- Any of the complex data structures (Array and Objects)

WHAT VARIABLE IS IN JS?

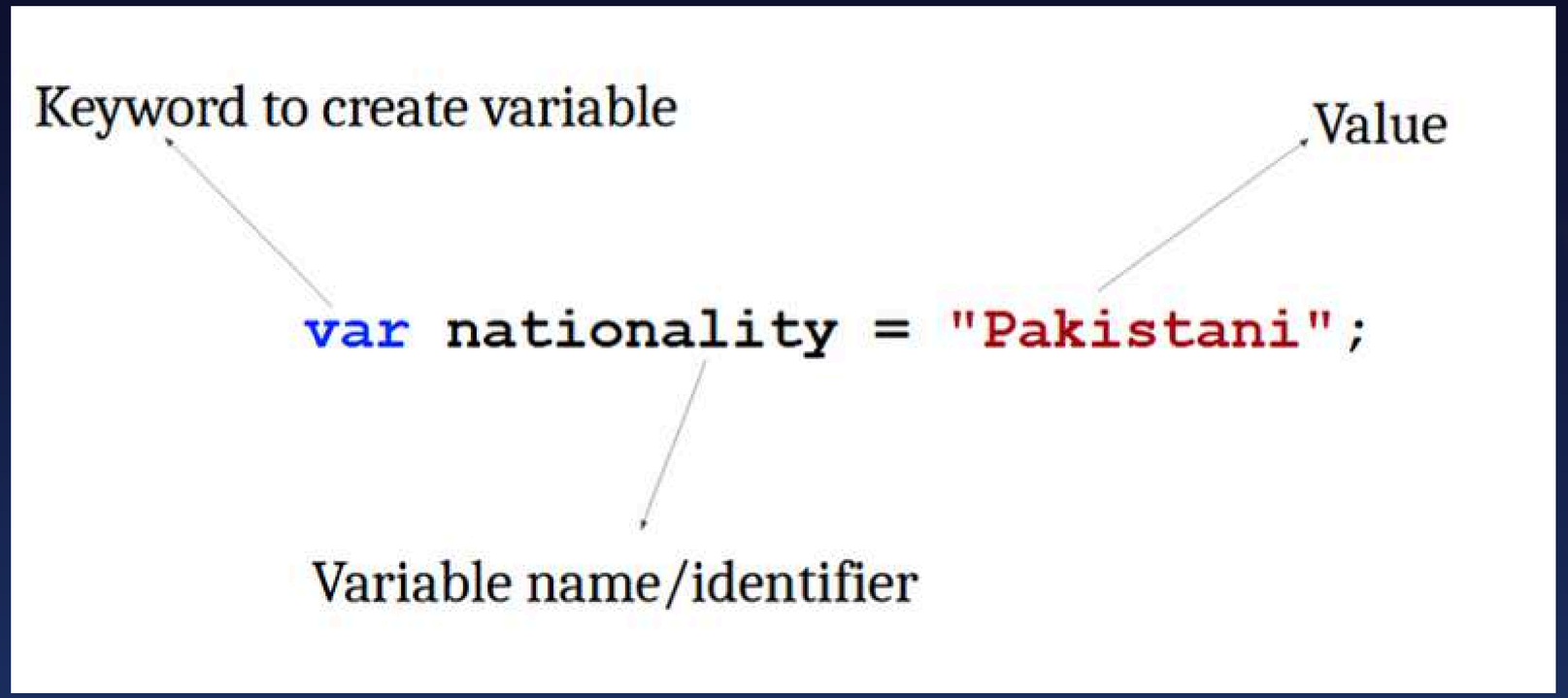
There are 3 ways to declare a JavaScript variable:

- Using var
- Using let
- Using const

VARIABLES IN JS (VAR, LET & CONST)?

- var and let create variables that can be reassigned another value.
- const creates "constant" variables that cannot be reassigned another value.
- developers shouldn't use var anymore. They should use let or const instead.
- if you're not going to change the value of a variable, it is good practice to use const.

VARIABLES


Keyword to create variable
Value
var nationality = "Pakistani";
Variable name/identifier

VARIABLES

```
var nationality = "Pakistani";

var age = 25;

var isFeePaid = true;

var weight = 60.55;
```

DECLARATION AND INITIALIZATION

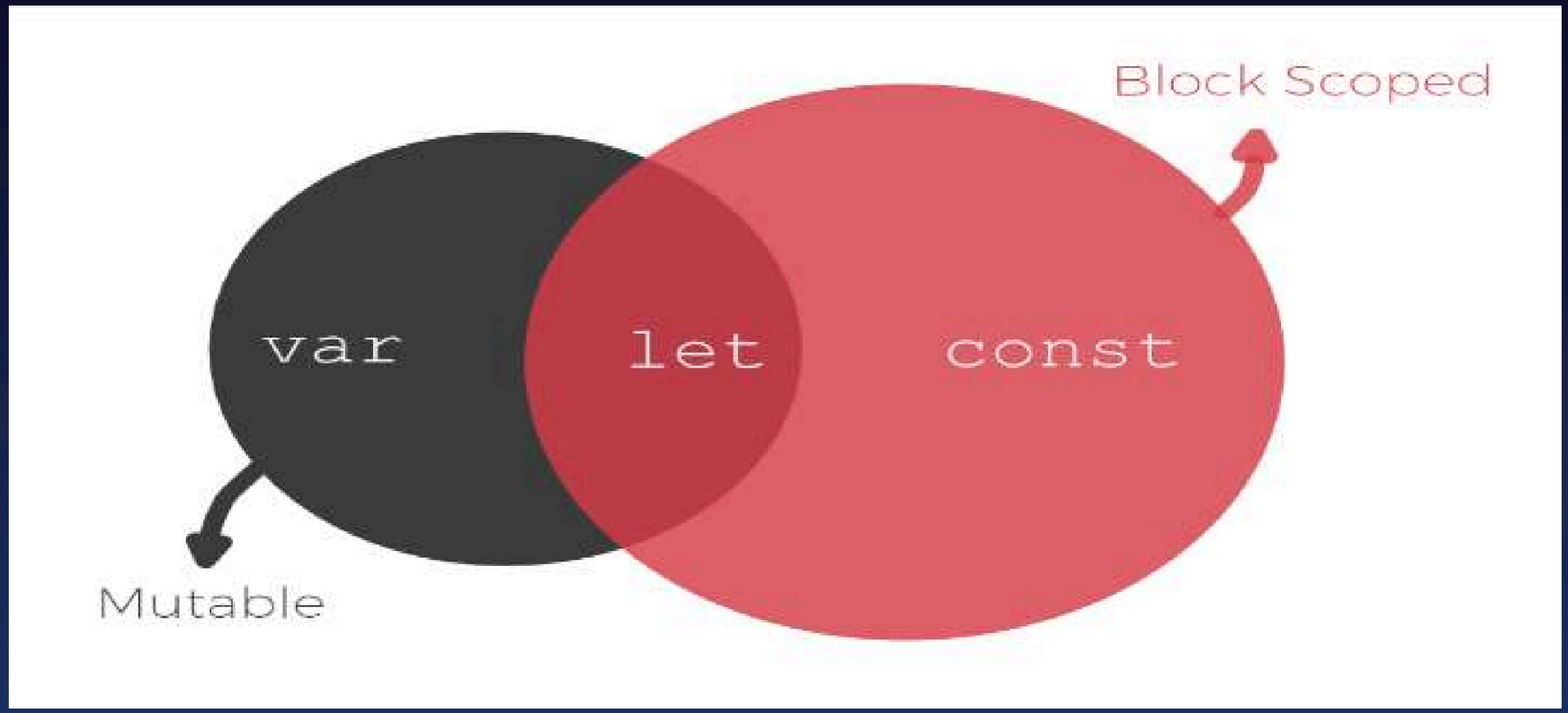
1. You can declare and initialize in single line or you can do that in two line

```
var  age = 25;   OR

var  age;     -- Declaration
age  = 25;    -- Initialization
```

Only declaration will leave variable undefined

VARIABLES' TYPES

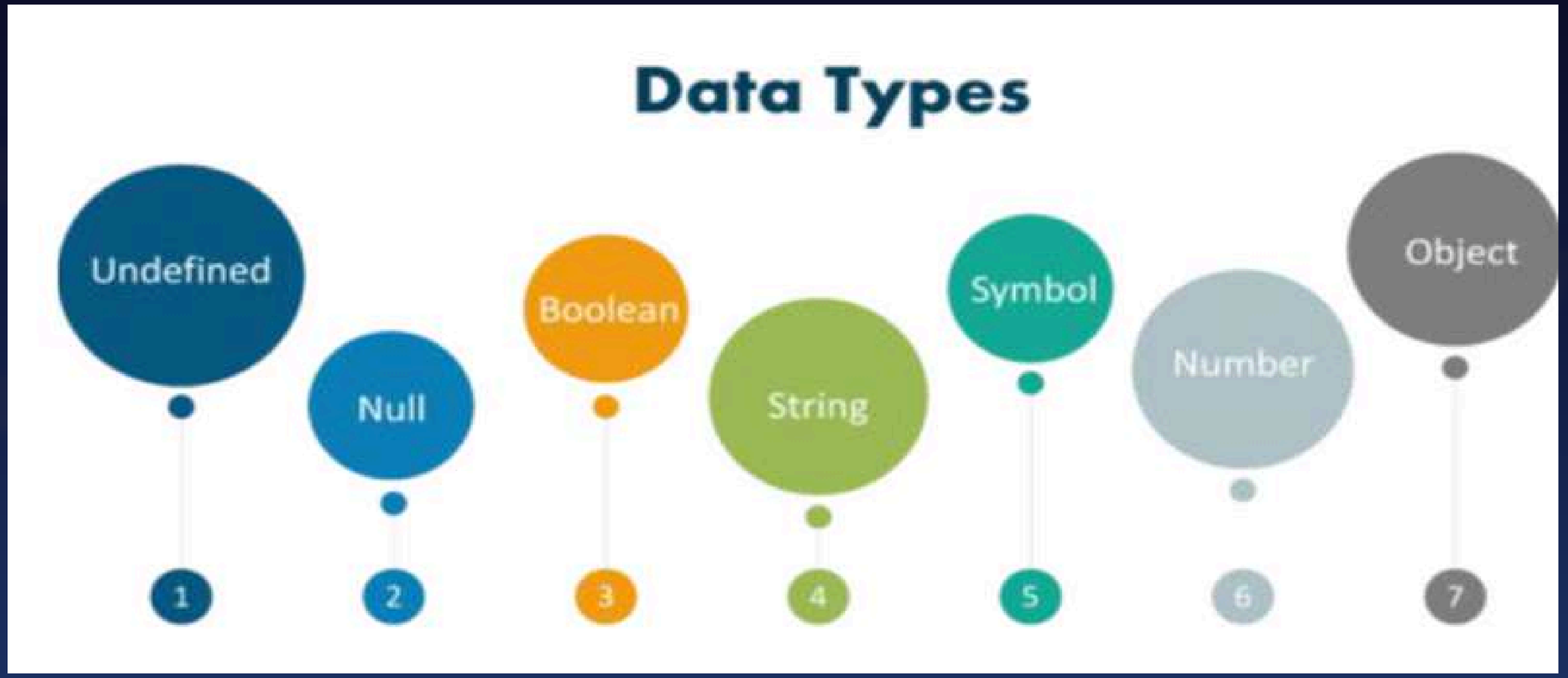


VARIABLES' TYPES

var	let	const
The scope of a *var* variable is functional scope.	The scope of a *let* variable is block scope.	The scope of a *const* variable is block scope.
It can be updated and re-declared into the scope.	It can be updated but cannot be re-declared into the scope.	It cannot be updated or re-declared into the scope.
It can be declared without initialization.	It can be declared without initialization.	It cannot be declared without initialization.
It can be accessed without initialization as its default value is "undefined".	It cannot be accessed without initialization otherwise it will give 'referenceError'.	It cannot be accessed without initialization, as it cannot be declared without initialization.

DATA TYPES IN JAVASCRIPT

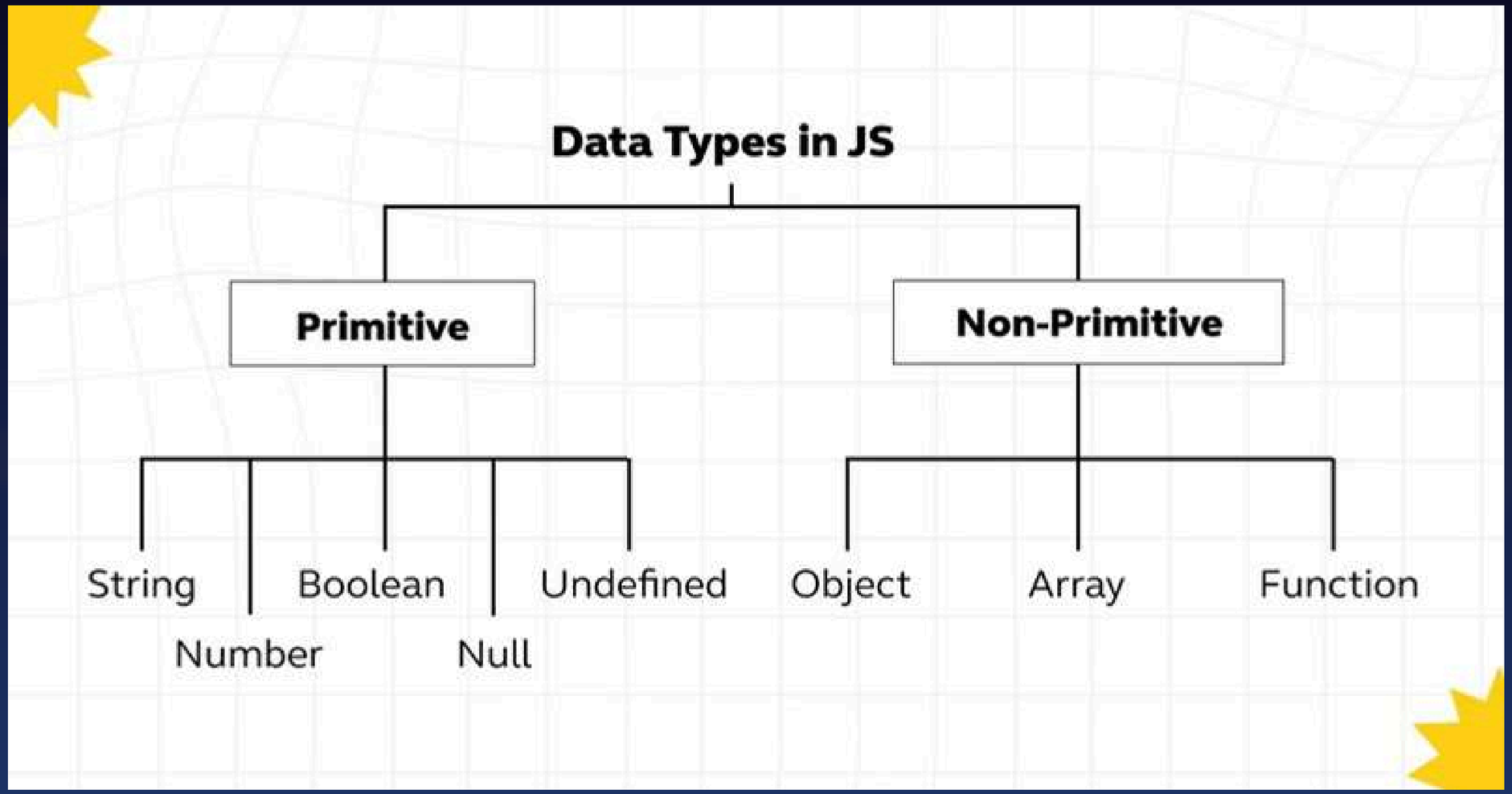
DATA TYPES


Data Types
Undefined
Null
Boolean
String
Symbol
Number
Object
1
2
3
4
5
6
7

DATA TYPES

- Datatypes hold different values.
- There are two types of datatypes in JavaScript: Primitive and Non-Primitive.
- Primitive defines immutable values and was introduced recently by ECMAScript standard.

DATA TYPES


Data Types in JS
Primitive
Non-Primitive
String
Number
Boolean
Null
Undefined
Object
Array
Function

DATA TYPES

- Datatypes hold different values.
- There are two types of datatypes in JavaScript: Primitive and Non-Primitive.
- Primitive defines immutable values and was introduced recently by ECMAScript standard.

DATA TYPES

Six types are considered to be **primitives.**

- Number -- integers, floats etc
- String -- an array of characters
- Boolean -- true or false
- Null -- No Value
- Undefined --a declared variable but hasn’t been given a value
- Symbol -- a unique value that's not equal to any other value

DATA TYPES

Non-primitive / Complex data Types

- Objects
- Functions

Function is callable object that executes a block of code, it lies under the object type. Non-primitive data types is the object. The JavaScript arrays and functions are also objects.

DATA TYPES

The next basic data type is the string. Strings are used to represent text. They are written by enclosing their content in quotes.

To store text in variable we use **strings**

```
var name = "Rizwan";
name = "Saleem";
```

DATA TYPES

Strings

- A string variable can store a sequence of alphanumeric characters, spaces and special characters.
- Each character is represented using 16 bit
  — You can store Chinese characters in a string.
- A string can be enclosed by a pair of single quotes (') or double quote (").
- Use escaped character sequence to represent special character (e.g.: \", \n, \t)

STRING - SINGLE QUOTES AND DOUBLE QUOTES

- Text strings are always enclosed in quotes
- You can use single or double quotes

```
var nationality = "Pakistani";
var message = 'Where is your Passport?';
```

VARIABLE FOR NUMBERS

- To store whole numbers or decimal values we use number data type

```
var age = 25;
var weight = 150.5
```

VARIABLE FOR BOOLEAN

- It is often useful to have a value that distinguishes between only two possibilities, like “yes” and “no” or “on” and “off”. For this purpose, JavaScript has a Boolean type, which has just two values, true and false, which are written as those words.

```
var isFeePaid = true;
var examPassed = false;
```

UNDEFINED

- The undefined data type can only have one value-the special value undefined
- If a variable has been declared, but has not been assigned a value, has the value **undefined**

```
var name;
var age;
```

NULL

- This is another special data type that can have only one value-the null value.
- A null value means that there is no value.
- It is not equivalent to an empty string ("") or 0, it is simply nothing

```
var name = null;
var nationality = "Pakistani";
nationality = null;
```

DIFFERENCE BETWEEN NULL AND UNDEFINED

- undefined means a variable has been declared but has not yet been assigned a value.
- null is an assignment value. It can be assigned to a variable as a representation of no value
- undefined and null are two distinct types

  undefined is a type itself (undefined) while null is an object.
- Unassigned variables are initialized by JavaScript with a default value of undefined. JavaScript never sets a value to null. That must be done programmatically.

DIFFERENCE BETWEEN NULL AND UNDEFINED

- When we define a variable to undefined then we are trying to convey that the variable does not exist .
- When we define a variable to null then we are trying to convey that the variable is empty.

STATEMENTS AND EXPRESSIONS

STATEMENTS

- A computer program is a list of "instructions" to be "executed" by a computer.
- In a programming language, these programming instructions are called statements.
- A JavaScript program is a list of programming statements.
- A statement can set a variable equal to a value.
- A statement can also be a function call, i.e.
- document.write().

STATEMENTS

- Statements define what the script will do and how it will be done.
- Most JavaScript programs contain many JavaScript statements.
- The statements are executed, one by one, in the same order as they are written.

STATEMENTS

- Statements define what the script will do and how it will be done.
- Most JavaScript programs contain many JavaScript statements.
- The statements are executed, one by one, in the same order as they are written.

STATEMENTS

- Each line below is a statement

```
var a = 4;              // Statement 1
var b = 2;              // Statement 2
var c = 0;              // Statement 3
c = a + b;              // Statement 4

alert(a);               // Statement 4
console.log(c);         // Statement 4
```

END OF STATEMENT WITH SEMICOLON ;

- To end statement add semicolon at the end of each executable statement
- Semicolons separate JavaScript statements.

```
var a = 4;
```

- When separated by semicolons, multiple statements on one line are allowed

```
i = 3; j = 5; k = i + j;

alert(i); console.log(k);
```

-

END OF STATEMENT WITH SEMICOLON ;

- Typically we end statements with semicolon but JavaScript semicolon is optional, but is highly recommended to end with semicolon
- The end of a statement is most often marked by pressing enter and starting a new line.

END OF STATEMENT WITH SEMICOLON ;

```
var a = 5                    // New line will end statement
a * 4
alert(a)
console.log(a)



var a = 5a * 4               // Error, Will Not work
alert(a)console.log(a)       // Error, Will not work
```

END OF STATEMENT WITH SEMICOLON ;

- An expression is a combination of values, variables, function calls and operators, which computes to a value.

var a + b;	Statement
var c = 5;	Statement
var d = “Hello world”	Statement

COMMENTS

- Comments are for the human, not the machine.
- They help you and others understand your code when it
- comes time to revise, they make code more readable.
- You can also use commenting to comment out portions of your code for testing and debugging.
- They will prevent execution of code
- There are two ways mark text as a comment, single line
- comments and multi-line comments

COMMENTS

- Comments are for the human, not the machine.
- They help you and others understand your code when it
- comes time to revise, they make code more readable.
- You can also use commenting to comment out portions of your code for testing and debugging.
- They will prevent execution of code
- There are two ways mark text as a comment, single line
- comments and multi-line comments

SINGLE LINE COMMENTS

Single line comments start with //.

Any text between / and the end of the line will be ignored (will not be executed).

```
// Declare and initialize variable var a = 6;

//This is comment

//This below code will not execute

var b = 8;
```

MULTI LINE COMMENTS

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored.

```
/*Declare and initialize variable var a = 6;
This is comment
This below code will not execute*/
var b = 8;
```

VARIABLE NAMES

VARIABLE NAMES LEGAL AND ILLEGAL

- A variable name can't contain any spaces
- A variable name can contain only letters, numbers dollar signs, and underscores.
- The first character must be a letter, or an underscore (_), or a dollar sign ($).
- Subsequent characters may be letters, digits, underscores, or dollar signs.
- Numbers are not allowed as the first character of variable.

VARIABLE NAMES LEGAL AND ILLEGAL

1. Legal names:

```
var hello   =   56;
var _xyz    =  44;
var $work   =   90;
var user2   =   56;
var i_info = 99;
var my$wor = 77;
```

VARIABLE NAMES LEGAL AND ILLEGAL

1. Illegal names:

```
var 2user = 12; // Can't start with number
var my user = 23; // Can't contains space
var hello#world = 34;
var   my-info = 44;
var my?info = 45;
var my*info = 45;
```

RESERVED KEYWORDS

- Reserved Keywords cannot be used as variable name
- Here are few reserved keywords

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*

RESERVED KEYWORDS

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

VARIABLE NAMES LEGAL AND ILLEGAL

- A variable name can't contain any spaces
- A variable name can contain only letters, numbers dollar signs, and underscores.
- The first character must be a letter, or an underscore (_), or a dollar sign ($).
- Subsequent characters may be letters, digits, underscores, or dollar signs.
- Numbers are not allowed as the first character of variable.

CASE SENSITIVE

- Variable names are case sensitive.
- So rose and Rose are two different variables

```
var rose = "Hello";
var Rose = "Hello";
 alert(rose);
 alert(Rose);
 alert(ROSE); // Error
```

CAMEL CASE

- If there's more than one word in the variable name, then it is recommended to use camel case
- A camelCase name begins in lower case. If there's more than one word in the name, each subsequent word gets an initial cap, creating a hump.

```
var userResponse
var userResponseTime
var userResponseTimeLimit
var response
```

This Style of naming a variable is called camel case

OPERATORS

ARITHMETIC OPERATORS

Arithmetic operators perform arithmetic on numbers (literals or variables).

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

ARITHMETIC OPERATORS

```
let num1 = 20;
let num2 = 10;

let sum = num1 + num2;        // Addition
let difference = num1 - num2;  // Subtraction
let product = num1 * num2;     // Multiplication
let quotient = num1 / num2;    // Division
let remainder = num1 % num2;   // Modulus
let power = num1 ** 2;         // Exponentiation

console.log("Sum: " + sum);
console.log("Difference: " + difference);
console.log("Product: " + product);
console.log("Quotient: " + quotient);
console.log("Remainder: " + remainder);
console.log("Power: " + power);
```

ARITHMETIC OPERATORS

Operator	Name	Purpose	Example
`+`	Addition	Adds two numbers together.	`6 + 9`
`-`	Subtraction	Subtracts the right number from the left.	`20 - 15`
`*`	Multiplication	Multiplies two numbers together.	`3 * 7`
`/`	Division	Divides the left number by the right.	`10 / 5`
`%`	Remainder (sometimes called modulo)	Returns the remainder left over after you've divided the left number into a number of integer portions equal to the right number.	`8 % 3` (returns 2, as three goes into 8 twice, leaving 2 left over).
`**`	Exponent	Raises a `base` number to the `exponent` power, that is, the `base` number multiplied by itself, `exponent` times. It was first introduced in EcmaScript 2016.	`5 ** 2` (returns `25`, which is the same as `5 * 5`).

ASSIGNMENT OPERATORS

- Assignment operator assign value to variables
- When you need to apply arithmetic operation and assign value to same variable then you can also use

```
var a = 5; // equals = is assignment operator
a = a + 2; // Assign 7 in variable a
OR
var a = 5;
a+=2;      // Assign 7 in variable a
```

ASSIGNMENT OPERATORS

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y
**=	x **= y	x = x ** y

ELIMINATING AMBIGUITY -- BODMAS

In JavaScript (and most programming languages), arithmetic operations follow BODMAS (Brackets, Orders (exponents), Division and Multiplication, Addition, and Subtraction) to eliminate ambiguity. This ensures that calculations are performed in the correct order.

```
let result = (10 + 5) * 2 - 3 ** 2 / 3;

console.log(result);
```

Breaking it down (Step-by-Step Execution):

1. Brackets: `(10 + 5) = 15`
2. Orders (Exponentiation): `3 ** 2 = 9`
3. Division: `9 / 3 = 3`
4. Multiplication: `15 * 2 = 30`
5. Subtraction: `30 - 3 = 27`

ELIMINATING AMBIGUITY -- BODMAS

```
var a = 3 + 5 * 2;                    // result 13
var b = 8 / 2 - 1;                    // result 3
var c = 3 % 2 + 4  - 1;               // result 4
```

THANK-YOU