<script>
document.getElementById("demo").innerHTML = "John Doe";
</script>

o/p:- John Doe.

Javascript variables:-

→ variables are used to store data values.
→ Javascript uses the keywords var, let and const to declare variables.
→ An equal sign is used to assign values to variables.

```
let x;
x=6;
```

→ Global variables:- can be declared outside the function or declared with window object.

```
<script>
var value=50;
function a() {
alert(value);
}
function b() {
alert(value);
}
</script>
```

Javascript operators:-

1. arithemetic (+, -, *, /)
2. assignment operator

Javascript Expressions:-

→ An expression is a combination of values, variables and operators, which computes a value.
→ The computation is called an evaluation.
→ 5*10
→ Expressions can also contain variable values:-

X * 10

"John" + " " + "Doe" ⇒ John Doe.

Javascript keywords:-

→ let keyword tells the browser to create variables.
→ The var also tells browser to create variables

→ Identifiers are JavaScript names.
→ Identifiers are used to name variables, keywords, and functions.
→ The rules for legal names are the same in most programming languages.
JavaScript name must begin with:-
A letter (A-Z or a-z)
A dollar sign ($)
or an underscore (_)
→ subsequent characters may be letters, digits, underscores, or dollar signs.
→ Numbers are not allowed as the first character in names.
→ This way JavaScript can easily distinguish identifiers from numbers.

JavaScript is case sensitive:

→ Identifiers are case-sensitive.
→ JavaCode does not interpret LET or Let as the keyword let.
→ Hyphens are not allowed in JavaScript. They are reserved for subtractions.
→ Underscore:-

```
first_name
last_name
```

→ Upper Camel Case (Pascal)

```
FirstName
LastName
```

→ lower camel case:-

```
firstName
lastName
```

JavaScript character set:-

JavaScript uses the Unicode character set.
Unicode covers (almost) all the characters

→ from 1995 to 2015.
→ The let and const keywords were added to Javascript in 2015.
→ The var keyword should only be used in Code written for old browsers.
→ Javascript variables can hold numbers like 100 and text values like "John Doe".
→ In programming, text values are called text strings.
→ Strings are written inside double or single quotes. Numbers are written without quotes.
→ If you put a number in quotes, it will be treated as a text string.
→ Variables declared with let cannot be redeclared in the same scope.
→ Javascript did not have block scope.
→ Javascript had Global scope & Function scope
Variables declared inside a {} block cannot be accessed the block.

```
{
let x=2;
}
```

Global scope:-
→ Variables declared with the var always have global scope.
→ Variables declared with the var keyword Can Not have block scope:-
→ Variables declared with var inside a {} block can be accessed from outside the block.

```
{
var x=2;
}
var x=2; // now x is 2.
```

```
{ var x=2;
  let x=3; // not allowed
}
{ let x=2;   allowed
  let x=3;   not allowed
}
{
  let x=2; // Allowed
  var x=3; // Not allowed
}
```

```
<script>
function abc() {
  var x=10;
}
</script>
```
→ local variable.

```
let x=2;
{
  let x=3; // Allowed
}
let x=4; // Allowed
```

```
<script>
var data=200; // global variable
function a() {
document.writeln(data)
}
function b() {
document.writeln(data);
}
```

	scope	redeclare	reassign	Hoisted	Bind this
var	No	yes	yes	yes	yes
let	yes	No	yes	No	No
Const	yes	No	No	No	No

Javascript operators:

These are different types of Javascript operators:-

1. Arithematic
2. Assignment
3. comparison
4. string
5. Logical
6. Bitwise
7. Ternary

→ var allows redeclaration and is not block-scoped.

→ let is block-scoped and cannot be redeclared in the same scope.

→ Const is block-scoped, cannot be reassigned, but objects declared with const can have their properties

+, -, *, **(Exponetion)
/ division
% modulus
++ Increment
-- decrement

Assignment:- Assigns vallue
=, +=, -=, *=, /=, %=, **=

Comparision:-
==, [=== [equal value & equal type)]
!= [NOT equal)
[!== [notequal value or not equal type)]
> [greater than)
< less than)
>= greater than or equal to
<= less than or equal

? ternary

Logical:-
&& - and
|| - OR
! - NOT

Bitwise:-
& - Bitwise AND
| - Bitwise OR
^ - Bitwise x-OR
~ - Bitwise NOT
<< - Bitwise left shift
>> - Bitwise right shift
[>>> - Bitwise right shift with zero]

Type operators:-
typeof - Returns the type of a variable
instance of - returns true if an object is an instance of an object type.

Bitwise:-
&, |, ~, ^, <<, >>
↓ ↓
left shift. Right Shift

>>>:-

is used between two values.

→ If the first value is ~~defined~~ undefined or null, the second value is assigned.

```
let x;
x ??= 5;
```

→ If the variable is null or undefined right hand side value is assigned

→ If the variable already has a defined it remains unchanged.

```
let a=null;
a ??=10; → assigns 10 because a is null
Console.log(a)
let b=5;
b ??=10; → b is already defined
console.log(b); → 115
```

Data types:-

→ String
→ Number
→ bigint
→ undefined
→ Null
→ symbol
→ object → It contains An object
An array
A date

objects:-

```
const person= {firstName: "John", lastName
: "Doe"};
```

//Array object:-

```
const cars= ["saab", "volvo", "BMW"];
```

//Date object:-

```
const date= new Date ("2022-03-25");
```

Javascript types are Dynamic:-

```
let x;
x=5;
x="John";
```

Javascript special operators:- (?:) → Ternary operator

void → discards the expression's return value.

Comma → allows multiple expressions to be evaluated as single statement

delete - delete operator deletes a property from the object.

ned to perform a particular task.
→ A JavaScript function is a block of
→ A JavaScript function is executed when something call it.
→

```
<p id="demo"></p>
<script>
function myFunction(p1, p2) {
    return p1*p2;
}
let result = myFunction(4,3);
document.getElementById("demo").innerHTML = result;
</script>
```

→ Function syntax:
→ JavaScript function is defined with function keyword, followed by parentheses().
→ Function name can contain letters, digits, underscore and dollar signs.
→ the parentheses include parameter names separated by commas:- (parameter1, parameter2 -),
→ code must be in curly braces.
→

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

→ Function parameters are listed inside the parenthesis() in the function definition
→ Function arguments are the values received by the function when it is invoked.
→ Inside the function, the arguments behave as local variables.

```
<!doctype html>
                                    </head>
                                    <body>
                                    <script type="text/
```

a button)
→ When it is invoked (called) from JavaScript code.
→ Automatically (self invoked)

Function return:-

→ When JavaScript reaches a return statement, the function will stop executing.
→ If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
→ Function often compute a return value. The return value is "returned" BACK to "caller".
→ The () operator:-
It calls the function.
→ Local variables.

```
let x = myFunction(4,3); // function is called
                         the return value will end up in x
function myFunction(a,b) {
    // function returns the product of a & b
    return a*b;
}
```

JavaScript objects:- JavaScript is an object-based language. Everything is an object in JavaScript. ↗ JavaScript is template based not class based.

Real life objects, properties and methods:-

→ In real life car is an object. → Here we don't create object to get the object.
→ A car has properties like weight, color & methods like start and stop. But we direct create objects.

properties	methods
car.name = Fiat	car.start()
= 500	= car.brake()

```
}
document.write(carmodel
(addTest());
```

property values differ from car to car.
→ All cars have the same methods, but the methods are performed at different times.

JavaScript Objects():

This code assigneds a simple value (Fiat) to a variable named car:-

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript</h2>
<p id="demo"></p>
<script>
let car= "Fiat";
document.getElementById("demo").innerHTML
=car;
</script>
</body>
</html>
```

3 objects in JavaScript
1. By using literal
2. By creating instance of object directly (using new keyword).
3. By using an object constructor (using new keyword).

Syntax for object:
object= {property1: value1, property2: value2 ... propertyN: valueN}

JavaScript variables
Fiat

→ objects are variables too, but objects can contain many values!
→ This code assigns many values (Fiat, 500, white) to a variable named car.

```
Const car= {type: "Fiat", model: "500",
            color: "white"};
```

object definition:
You define a JavaScript object with an object literal.

```
const person= {firstname: "John",
```