ADDING COMMENTS IN JAVASCRIPT

1.Single-line Comment

- Use // to add a comment on a single line.

```
// This is a single-line comment
```

2.Multi-line Comment

- Use /* */ for comments spanning multiple lines.

```
/*
 This is a
 multi-line comment
*/
```

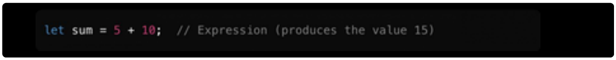
Example:

```
// This is a single-line comment
let x = 5;  // This comment is at the end of a line

/*
 This is a multi-line comment
 explaining the following code
*/
let y = 10;
```

EXPRESSIONS IN JAVASCRIPT AND THEIR DIFFERENCE FROM STATEMENTS

1.Expression

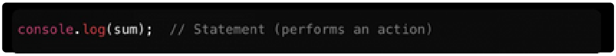
- An expression produces a value (e.g., arithmetic operations or variable assignments).



```
let sum = 5 + 10;  // Expression (produces the value 15)
```

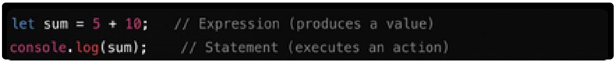
2.Statement

- A statement performs an action (e.g., variable declaration, function call).



```
console.log(sum);  // Statement (performs an action)
```

Example:



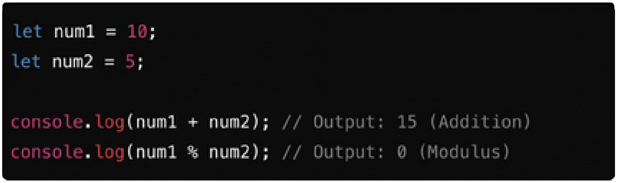
```
let sum = 5 + 10;    // Expression (produces a value)
console.log(sum);     // Statement (executes an action)
```

JAVASCRIPT OPERATORS

Arithmetic Operators

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Division)
- % (Modulus – Remainder of division)

Example:



```
let num1 = 10;
let num2 = 5;

console.log(num1 + num2); // Output: 15 (Addition)
console.log(num1 % num2); // Output: 0 (Modulus)
```

INCREMENT & DECREMENT OPERATORS

Increment (++)

- Increases the value of a variable by 1.

```
count++; // Increment
```

Decrement (--)

- Decreases the value of a variable by 1.

```
count--; // Decrement
```

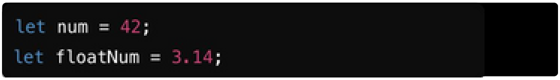
Example:

```
let count = 5;
count++;             // Increment
console.log(count); // Output: 6
```

PRIMITIVE DATA TYPES IN JAVASCRIPT

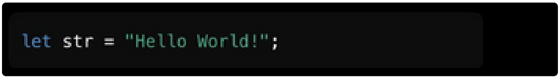
Primitive data types are immutable and stored directly in memory.

1.Number (Integer & Floating-point)



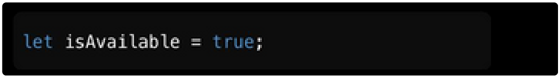
```javascript
let num = 42;
let floatNum = 3.14;
```

2.String (Text enclosed in quotes)



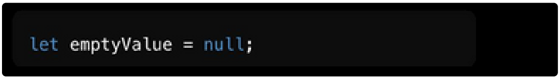
```javascript
let str = "Hello World!";
```

3.Boolean (Represents true or false)



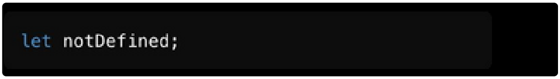
```javascript
let isAvailable = true;
```

4.Null (Represents an empty or non-existent value)



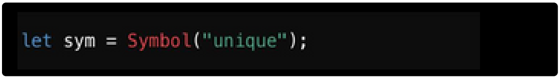
```javascript
let emptyValue = null;
```

5.Undefined (Declared but not assigned a value)



```
let notDefined;
```

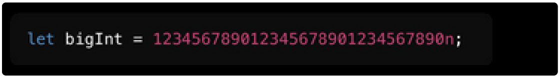
6.Symbol (Unique and immutable value)



```
let sym = Symbol("unique");
```

7.BigInt (Handles large numbers beyond

Number.MAX_SAFE_INTEGER)

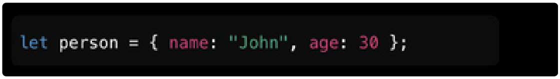


```
let bigInt = 1234567890123456789012345678901n;
```

REFERENCE (RELATIVE) DATA TYPES IN JAVASCRIPT

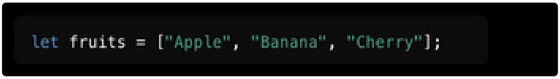
Reference data types are stored in memory by reference and can be modified.

1.Object (Collection of key-value pairs)



```
let person = { name: "John", age: 30 };
```

2.Array (Ordered list of values)



```
let fruits = ["Apple", "Banana", "Cherry"];
```

JAVASCRIPT DATA TYPES

JavaScript has 8 data types, categorized into Primitive and Reference types.

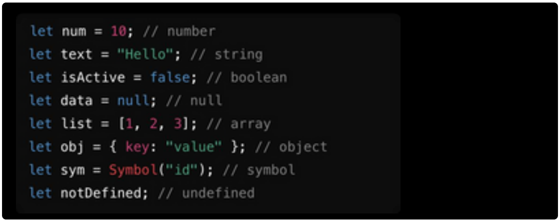
Primitive Data Types:

- Number → let num = 10
- String → let text = "Hello"
- Boolean → let isActive = false
- Null → let data = null
- Undefined → let notDefined
- Symbol → let sym = Symbol("id");

Primitive Data Types:

- Array → let list = [1, 2, 3]
- Object → let obj = { key: "value" };

Example:

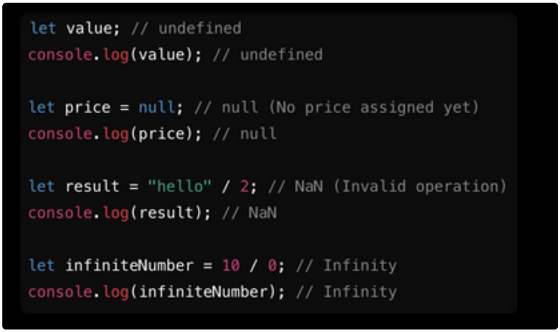


```
let num = 10; // number
let text = "Hello"; // string
let isActive = false; // boolean
let data = null; // null
let list = [1, 2, 3]; // array
let obj = { key: "value" }; // object
let sym = Symbol("id"); // symbol
let notDefined; // undefined
```

SOME IMPORTANT VALUES IN JAVASCRIPT

1. undefined (Variable declared but not assigned a value)
2. null (Intentional absence of value)
3. NaN ("Not-a-Number" - Invalid mathematical operations)
4. Infinity (Represents an infinite value)

Example:

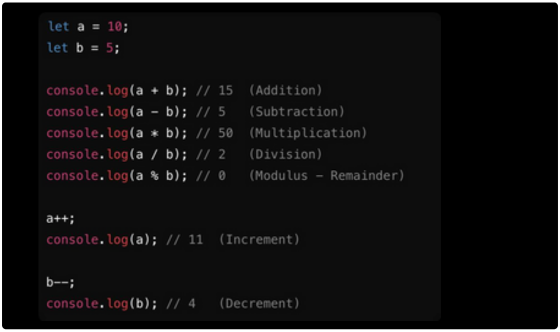


```
let value; // undefined
console.log(value); // undefined

let price = null; // null (No price assigned yet)
console.log(price); // null

let result = "hello" / 2; // NaN (Invalid operation)
console.log(result); // NaN

let infiniteNumber = 10 / 0; // Infinity
console.log(infiniteNumber); // Infinity
```

BASIC OPERATORS IN JAVASCRIPT

Arithmetic Operators (+,-,*,/,%,++,--)

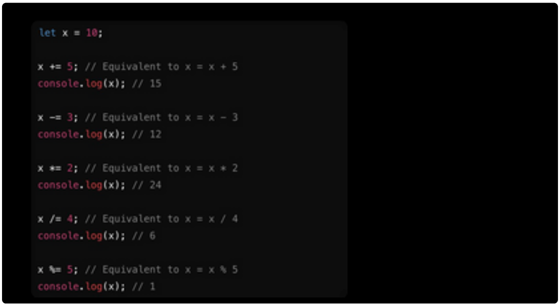
Example:



```
let a = 10;
let b = 5;

console.log(a + b); // 15  (Addition)
console.log(a - b); // 5   (Subtraction)
console.log(a * b); // 50  (Multiplication)
console.log(a / b); // 2   (Division)
console.log(a % b); // 0   (Modulus - Remainder)

a++;
console.log(a); // 11  (Increment)

b--;
console.log(b); // 4   (Decrement)
```

Assignment Operators (=,+=,-=,*=,/=,%=)

Example:

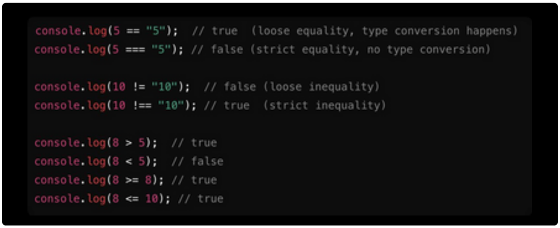


```
let x = 10;

x += 5; // Equivalent to x = x + 5
console.log(x); // 15

x -= 3; // Equivalent to x = x - 3
console.log(x); // 12

x *= 2; // Equivalent to x = x * 2
console.log(x); // 24

x /= 4; // Equivalent to x = x / 4
console.log(x); // 6

x %= 5; // Equivalent to x = x % 5
console.log(x); // 1
```

BASIC OPERATORS IN JAVASCRIPT

Comparison Operators (==, ===, !=, !==, >, <, >=, <=)

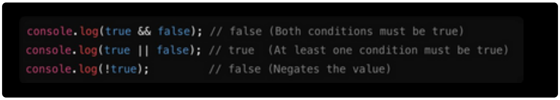
Example:



```
console.log(5 == "5");  // true  (loose equality, type conversion happens)
console.log(5 === "5"); // false (strict equality, no type conversion)

console.log(10 != "10");  // false (loose inequality)
console.log(10 !== "10"); // true  (strict inequality)

console.log(8 > 5);  // true
console.log(8 < 5);  // false
console.log(8 >= 8); // true
console.log(8 <= 10); // true
```

Logical Operators (&&, ||, !)

Example:

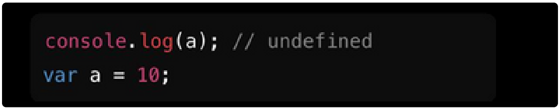


```
console.log(true && false); // false (Both conditions must be true)
console.log(true || false); // true  (At least one condition must be true)
console.log(!true);         // false (Negates the value)
```

VARIABLE HOISTING IN JAVASCRIPT

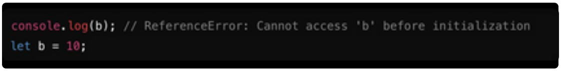
Hoisting moves variable and function declarations to the top of their scope before execution.

Using var (Hoisted but Undefined)



```javascript
console.log(a); // undefined
var a = 10;
```

Using let and const (Hoisted but Not Initialized)



```javascript
console.log(b); // ReferenceError: Cannot access 'b' before initialization
let b = 10;
```

Key Takeaway:

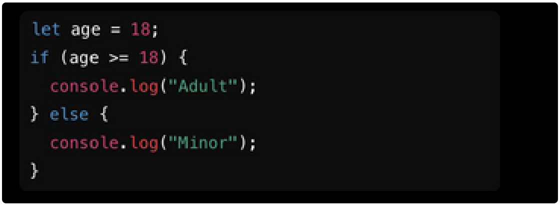
- var is hoisted with an initial value of undefined
- let and const are hoisted but remain in the Temporal Dead Zone until assigned.

CONDITION OPERATORS IN JAVASCRIPT

Hoisting moves variable and function declarations to the top of their scope before execution.

1.if-else Statement

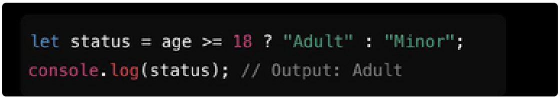
Used for Basic Conditional Checks.



```
let age = 18;
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```

2.Ternary Operator

A shorthand way of writing if-else statements



```
let status = age >= 18 ? "Adult" : "Minor";
console.log(status); // Output: Adult
```

Key Takeaway:

- if-else gives more flexibility for multiple conditions and is easy to read
- The ternary operator is great for simple conditions in a single line