**JAVASCRIPT**

1. Intro to Javascript

**JavaScript** is a lightweight, cross-platform, single-threaded, and interpreted compiled programming language. It is also known as the scripting language for webpages. It is well-known for the development of web pages, and many non-browser environments also use it.

* **Client-side:** It supplies objects to control a browser. Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as **mouse clicks**, **form input**, and **page navigation**. Useful libraries for the client side are [**AngularJS**](https://www.geeksforgeeks.org/introduction-to-angularjs/), [**ReactJS**](https://www.geeksforgeeks.org/react-js-introduction-working/), and so many others.
* **Server-side:** It supplies objects relevant to running JavaScript on a server. For if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is [**node.js**](https://www.geeksforgeeks.org/introduction-to-nodejs/).

1. **Basic structure of a js program**
2. **Errors and warnings**

# **JavaScript Operators:**

* Assignment Operators
* Arithmetic Operators
* Comparison Operators
* Logical Operators
* Bitwise Operators
* String Operators

## **JavaScript Assignment Operators**

|  |  |  |
| --- | --- | --- |
| = | Assignment operator | a = 7; // 7 |
| += | Addition assignment | a += 5; // a = a + 5 |
| -= | Subtraction Assignment | a -= 2; // a = a - 2 |
| \*= | Multiplication Assignment | a \*= 3; // a = a \* 3 |
| /= | Division Assignment | a /= 2; // a = a / 2 |
| %= | Remainder Assignment | a %= 2; // a = a % 2 |
| \*\*= | Exponentiation Assignment | a \*\*= 2; // a = a\*\*2 |

## JavaScript Arithmetic Operators

|  |  |  |
| --- | --- | --- |
| Operator | Name | Example |
| + | Addition | x + y |
| - | Subtraction | x - y |
| \* | Multiplication | x \* y |
| / | Division | x / y |
| % | Remainder | x % y |
| ++ | Increment (increments by 1) | ++x or x++ |
| -- | Decrement (decrements by 1) | --x or x-- |
| \*\* | Exponentiation (Power) | x \*\* y |

let x = 5;

let y = 3;

// addition

console.log('x + y = ', x + y); // 8

// subtraction

console.log('x - y = ', x - y); // 2

// multiplication

console.log('x \* y = ', x \* y); // 15

// division

console.log('x / y = ', x / y); // 1.6666666666666667

// remainder

console.log('x % y = ', x % y); // 2

// increment

console.log('++x = ', ++x); // x is now 6

console.log('x++ = ', x++); // prints 6 and then increased to 7

console.log('x = ', x); // 7

// decrement

console.log('--x = ', --x); // x is now 6

console.log('x-- = ', x--); // prints 6 and then decreased to 5

console.log('x = ', x); // 5

//exponentiation

console.log('x \*\* y =', x \*\* y);

## JavaScript Comparison Operators

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| == | **Equal to**: returns true if the operands are equal | x == y |
| != | **Not equal to**: returns true if the operands are not equal | x != y |
| > | **Greater than**: true if left operand is greater than the right operand | x > y |
| >= | **Greater than or equal to**: true if left operand is greater than or equal to the right operand | x >= y |
| < | **Less than**: true if the left operand is less than the right operand | x < y |
| <= | **Less than or equal to**: true if the left operand is less than or equal to the right operand | x <= y |

// equal operator

console.log(2 == 2); // true

console.log(2 == '2'); // true

// not equal operator

console.log(3 != 2); // true

console.log('hello' != 'Hello'); // true

const a = 3, b = 2;

console.log(a > b); // true

### JavaScript Logical Operators

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| && | **Logical AND**: true if both the operands are true, else returns false | x && y |
| || | **Logical OR**: true if either of the operands is true; returns false if both are false | x || y |
| ! | **Logical NOT**: true if the operand is false and vice-versa. | !x |

const x = 5, y = 3;

(x < 6) && (y < 5); // true

## JavaScript bitwise Operators

|  |  |  |
| --- | --- | --- |
| Operators | Name | Example |
| & | Bitwise AND | x & y |
| | | Bitwise OR | x | y |
| ^ | Bitwise XOR | x ^ y |
| ~ | Bitwise NOT | ~x |

**JavaScript Bitwise AND**

Bitwise AND & returns **1** if the corresponding bits of both operands are **1** else it returns **0**.

|  |  |  |
| --- | --- | --- |
| Operand 1 | Operand 2 | AND Operation |
| 0 | 0 | 0 & 0 is **0** |
| 0 | 1 | 0 & 1 is **0** |
| 1 | 0 | 1 & 0 is **0** |
| 1 | 1 | 1 & 1 is **1** |

In binary,

12 = 01100

25 = 11001

// Bitwise AND Operation of 12 and 25

00001100

& 00011001

---------

00001000 = 8 (In decimal)

**Note**: Converting **12** to 32-bit binary gives us 00000000000000000000000000001100 and **25** gives 00000000000000000000000000011001. However, we have removed the preceding zeros for simplicity.

### JavaScript String Operators

// concatenation operator

console.log('hello' + 'world');

let a = 'JavaScript';

a += ' tutorial'; // a = a + ' tutorial';

console.log(a);

**Data Types in JS:**

Primitive Data type

* 1. Number

Let num1= 10 // Int

Let num2 = 5.7 // floating point

* 1. String :

Let str=”hello”

Let str2= ‘any ’

* 1. Boolean data type

Let isCoding=true

Let isold= false

* 1. Null

Let a=null;

* 1. Undefined :variable not been assigned any value
  2. Bigint: let a=BigInt()

Non Primitive datatype:

object