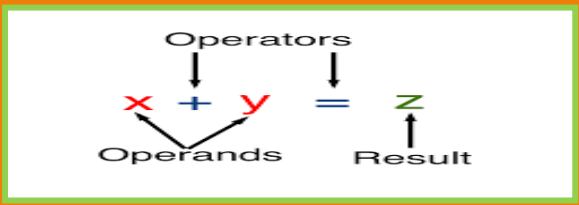
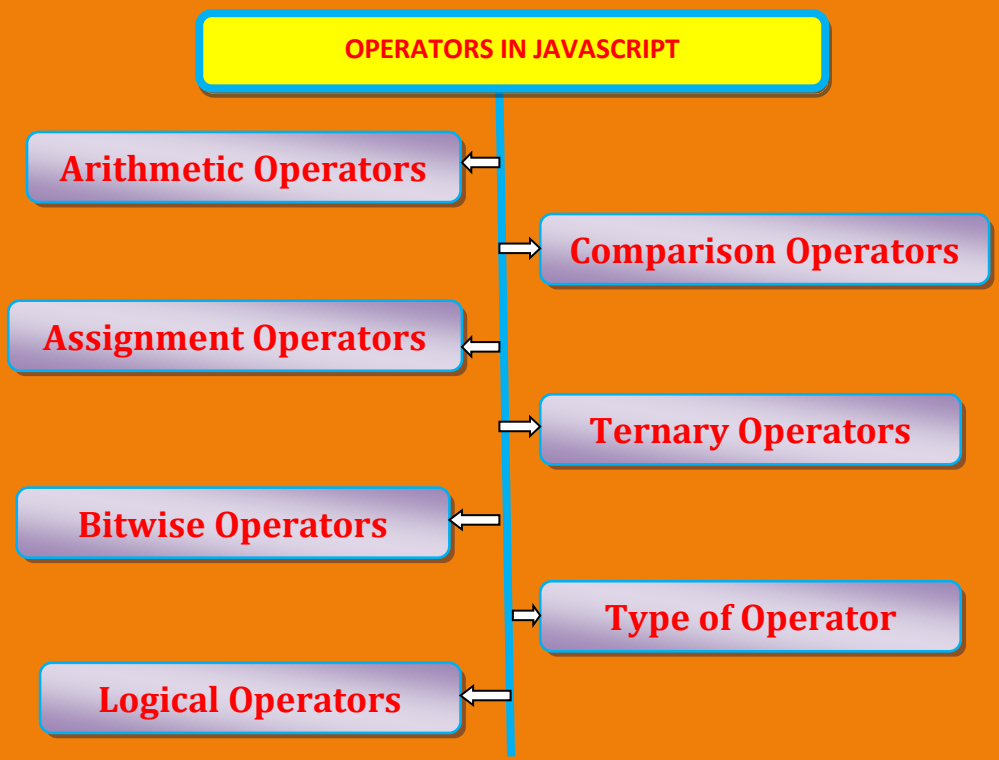
JAVASCRIPT NOTES

Operators

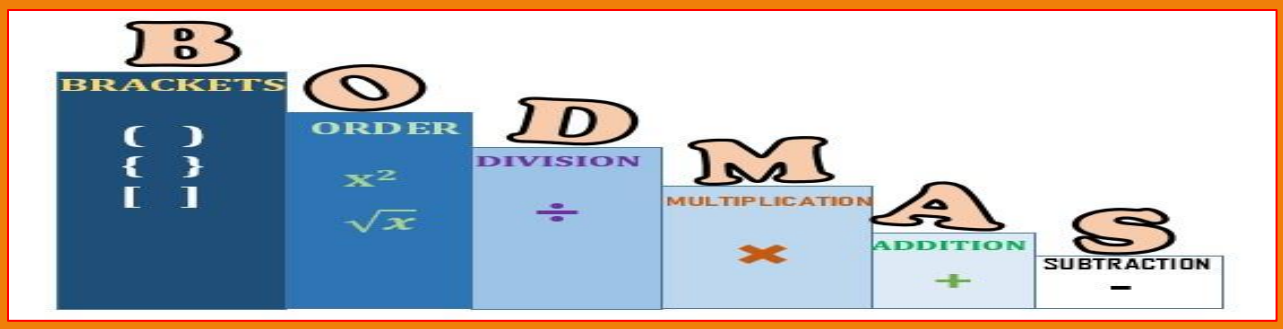
- Operators are special symbols used to assign values; compare values perform operations on single or multiple operands (data values) and produce the some result.
- A value or variable on which operators conduct operations is known as an operand.



- Different types of operators in JavaScript are as follows



Operators' precedence:



Arithmetic Operators

- **They perform an arithmetic operation between numerical variables and/or values. They include +, -, (*), (/), %, ++, --.**

The following math operations are supported:

- Addition `+`,
- Subtraction `-`,
- Multiplication `*`,
- Division `/`,
- Remainder `%`,
- Exponentiation `**`.

- **Examples for addition operators**

```
// addition operation between 2values
 let sum = 10 + 20;
console.log(sum); // 30
//addition operation between 2variables
let a1=10
let b1=20
let c1=a1+b1//30
```

```
// Converts non-numbers
alert( +true ); // 1
alert( +"" );   // 0
let apples = "2";
let oranges = "3"
//converted to num before the binary plus
alert( +apples + +oranges ); // 5
```

```
// addition operation (+)-is override for strings
let a=1+'3'//13-it converts 1 number to string implicitly
let b='1'+'3'//13-concates two strings
let c=NaN//NaN-not a number
let d=Infinity+Infinity//infinity
let e=3+ +'3'//6 -2nd + is unary plus operator( converts string to number)
alert(2 + 2 + '1' ); // "41" and not "221"
alert('1' + 2 + 2);//"122" and not "14"
```

Subtraction

The subtraction operator (-) subtracts numbers.

```
subtraction
alert( 6 - '2' ); // 4, converts '2' to a number
let sub=3-2//1
```

Multiplying

The multiplication operator (*) multiplies numbers.

```
multiplication
alert( 6 * '2' ); // 12, converts '2' to a number
let sub=3*2//6
```

Dividing

The division operator (/) divides numbers.

```
// divison-returns Quotient
alert( 6 / '2' ); // 3 , converts '2' to a number
let sub=3*2//1
```

Modulus

The modulus operator (%) returns the division remainder.

```
// modulus-returns reminder
alert( 6 / '2' ); // 0 , converts '2' to a number
let sub=3/2//1
```

Incrementing & Decrementing

The increment operator has 2 forms:

- **++a= pre-increment: increments the value by 1 before it used**
- **a++= post-increment : increments the value by 1 after it used**

- **--a= pre-decrement :decrements the value by 1 before it used**
- **a-- = post-decrement : decrements the value by 1 after it used**

```
// pre-increment and decrement operators
var a = 3;
// preincrement: 'a' is incremented before being assigned to 'ans'
var ans = ++a;
// postincrement: 'ans' is not incremented until after the line is printed.
console.log(ans++);
console.log(ans);
```

```
var a = 3;
// predecrement: 'a' is decremented before being assigned to 'ans'
var ans = --a;
// postdecrement: 'ans' is not decremented until after the line is printed.
console.log(ans--);
console.log(ans);
```

Exponentiation (introduced in ECMAScript 2016)

The exponentiation operator (**) raises the first operand to the power of the second operand.

```
let a = 2;
let b = b ** 2;
console.log(b)//4
```

```
var exp=2**2
console.log(exp)//4
var exp=2**'2'
console.log(exp)//4
var exp=2**'ggg'
console.log(exp)//nan
```

Assignment Operators

- **Assignment operators are used to assign values to variables**
- **= operator is used to assign value to variable**

Operator	Name	Example
`=`	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`

```
// var a=10
// var c=20
 c+=a//c+a-20+10-30
```

```
var str1='ravi'
str1+='kumar'//str1+kumar=ravikumar
```

Comparison Operators

Comparison operators compare two values and return a boolean value, either true or false.

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`
`===`	**Strict equal to:** `true` if the operands are equal and of the same type	`x === y`
`!==`	**Strict not equal to:** `true` if the operands are equal but of different type or not equal at all	`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(4 == 4); // true
console.log(4 == '4'); // true
// strict equal operator
console.log(4 === 4); // true
console.log(4 === '4'); // false
```

```
// not equal operator
console.log(3 != 2); // true
console.log(2 != 2); // false
console.log('hello' != 'Hello'); // true
// strict not equal operator
console.log(2 !== '2'); // true
console.log(2 !== 2); // false
```

```
// greater and lesser operator
console.log(4 > 4); // false
console.log(4 <= '4'); // true
```

Logical Operators

Logical operators perform logical operations and return a boolean value, either true or false

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`

A	B	A || B	A && B	!A
False	False	False	False	True
True	False	True	False	False
False	True	True	False	True
True	True	True	True	False

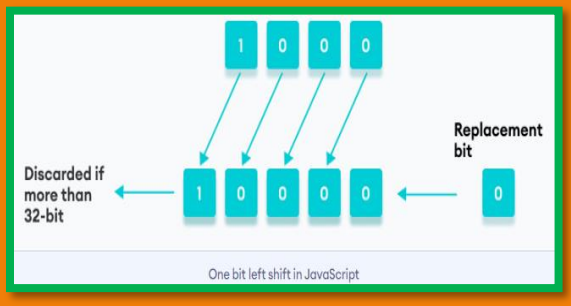
```
let b=20;
let d=!b//[b is true]
console.log(d)//false
```

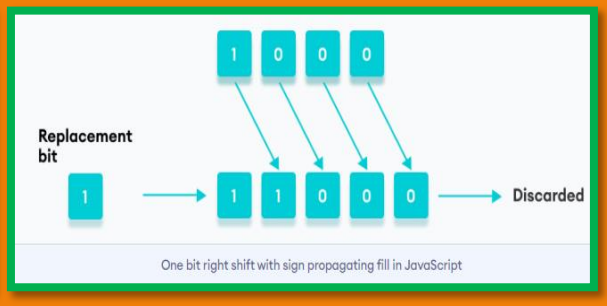
```
let a=20
let b=10
console.log(a>b&&a<b)//false
console.log(a>b||a<b)//true
console.log(!a )//false
```

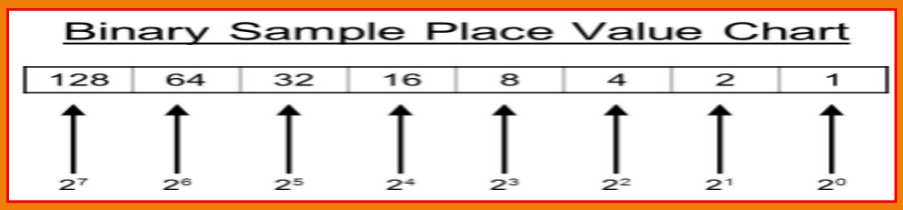
Bitwise Operators: **Bitwise operators perform operations on binary representations of numbers.**

- **JavaScript stores numbers as 64 bits floating point numbers, but all bitwise operations are performed on 32 bits binary numbers**
- **Before a bitwise operation is performed, JavaScript converts numbers to 32 bits signed integers**

Name	Symbol	Usage	What it does
Bitwise And	&	a&b	Returns 1 only if both the bits are 1
Bitwise Or	|	a|b	Returns 1 if one of the bits is 1
Bitwise Not	~	~a	Returns the complement of a bit
Bitwise Xor	^	a^b	Returns 0 if both the bits are same else 1
Bitwise Left shift	<<	a<<n	Shifts a towards left by n digits
Bitwise Right shift	>>	a>>n	Shifts a towards right by n digits







Number a = 25	0	0	0	1	1	0	0	1
Number b = 14	0	0	0	0	1	1	1	0
a & b	0	0	0	0	1	0	0	0

Number a = 25	0	0	0	1	1	0	0	1
~ a	1	1	1	0	0	1	1	0

Number a = 25	0	0	0	1	1	0	0	1
Number b = 14	0	0	0	0	1	1	1	0
a | b	0	0	0	1	1	1	1	1

Number a = 25	0	0	0	1	1	0	0	1
Number b = 14	0	0	0	0	1	1	1	0
a ^ b	0	0	0	1	0	1	1	1

Number a = 25	0	0	0	1	1	0	0	1
a << 3	1	1	0	0	1	0	0	0

Number a = 25	0	0	0	1	1	0	0	1
a >> 3	0	0	0	0	0	0	1	1

```
var d=200 //11001000
var f=d>>1// 01100100 // 100
console.log(f)//10
var e=f>>2
console.log(e) //00011001 //25
var g=e>>2
console.log(g)//00000011  //6
```

```
var d=200 //11001000
var f=d<<1// 10010000
console.log(f)//400
var e=f<<2
console.log(e) //1600
var g=e<<2
console.log(g)//6400
```

Ternary operator:

- A ternary operator is the short form If-else conditional statements
- A ternary operator evaluates a condition and executes a block of code based on the condition.

Syntax:

```
condition ? expression1 : expression2;
// If the condition is true, expression1 is executed.
// If the condition is false, expression2 is executed.
```

```
let age = 16;
let res =(age >= 18) ? "eligible to vote." : "not eligible to vote";
console.log(result);//not eligible to vote
```