💡 JavaScript Operators & Conditionals – Master Notes ✨

🧮 1. Arithmetic Operators

These operators perform basic math operations:

Operator	Name	Example	Output
`+`	Addition	`5 + 2`	`7`
`-`	Subtraction	`8 - 3`	`5`
`*`	Multiplication	`4 * 3`	`12`
`**`	Exponentiation	`2 ** 4`	`16`
`/`	Division	`10 / 2`	`5`
`%`	Modulus	`9 % 2`	`1`
`++`	Increment	`x++`	`x + 1`
`--`	Decrement	`x--`	`x - 1`

```
let increment = 5;
increment++; // 6

let decrement = 10;
decrement--; // 9
```

📝 2. Assignment Operators

Used to assign values and perform operations in shorthand:

Operator	Meaning	Example
`=`	Assign	`x = 5`
`+=`	Add and assign	`x += 3`
`-=`	Subtract and assign	`x -= 2`
`*=`	Multiply and assign	`x *= 4`
`/=`	Divide and assign	`x /= 2`
`%=`	Modulo and assign	`x %= 3`

Operator	Meaning	Example
`**=`	Exponent and assign	`x **= 2`

```
let x = 5;
x += 3; // 8
x -= 2; // 6
x *= 4; // 24
x /= 2; // 12
x %= 3; // 0
x **= 2; // 0
```

🔍 3. Comparison Operators

Used to compare values and return `true` or `false`:

Operator	Description	Example	Output
`==`	Equal (value)	`5 == '5'`	`true`
`!=`	Not equal (value)	`5 != 3`	`true`
`===`	Equal (value + type)	`5 === '5'`	`false`
`!==`	Not equal (value or type)	`5 !== '5'`	`true`
`>`	Greater than	`10 > 5`	`true`
`<`	Less than	`3 < 7`	`true`
`>=`	Greater than or equal to	`8 >= 8`	`true`
`<=`	Less than or equal to	`4 <= 2`	`false`

🧠 4. Logical Operators

Used to combine multiple boolean expressions:

Operator	Description	Example	Output			
`&&`	Logical AND	`true && false`	`false`			
`		`	Logical OR	`true	false`	`true`
`!`	Logical NOT	`!true`	`false`			

⚙️ 5. Bitwise Operators

Low-level binary operations:

```
let a = 5; // 0101
let b = 3; // 0011

console.log(a & b);    // 1   => 0001
console.log(a | b);    // 7   => 0111
console.log(a ^ b);    // 6   => 0110
console.log(~a);       // -6  => Inverts bits
console.log(a << 2);   // 20  => Shift left by 2 (5 * 4)
console.log(a >> 1);   // 2   => Shift right by 1 (5 / 2)
console.log(a >>> 1);  // 2   => Zero-fill right shift
```

💬 6. Comments in JS

```
// 👉 Single-line comment

/* 👇
Multi-line
comment
*/
```

🔁 7. Conditional Statements

✅ If Statement

```
let x = 10;
if (x > 5) {
  console.log("x is greater than 5"); // ✅ True
}
```

✅ If...Else Statement

```
let age = 18;
if (age >= 18) {
  console.log("You are eligible to vote.");
} else {
  console.log("You are not eligible to vote.");
}
```

✅ If...Else If...Else Statement

```javascript
let num = 5;
if (num > 0) {
  console.log("Number is positive.");
} else if (num < 0) {
  console.log("Number is negative.");
} else {
  console.log("Number is zero.");
}
```

🔀 8. Ternary Operator (Shorthand If...Else)

```javascript
let age = 18;
let message = (age >= 18) ? "You are an adult" : "You are not an adult";
console.log(message); // You are an adult
```

✅ **Example – Function with Ternary:**

```javascript
function checkEvenOrOdd(num) {
  return (num % 2 === 0) ? "Even" : "Odd";
}
console.log(checkEvenOrOdd(5)); // Odd
```

✅ **Nested Ternary (Be cautious!)**

```javascript
let n = 0;
let result = (n > 0) ? "Positive" : (n < 0) ? "Negative" : "Zero";
console.log(result); // Zero
```

➕ ➖ 9. Prefix vs Postfix Increment

```javascript
let a = 5;
let res1 = ++a; // 6 (increment first)
console.log(res1); // 6
console.log(a);    // 6

let b = 5;
let res2 = b++; // 5 (assign then increment)
console.log(res2); // 5
console.log(b);    // 6
```

🧪 10. Practice Questions (Conditional + Logical)

☑ Q1: Check if age is between 10 and 20

```
let age = 15;
if (age >= 10 && age <= 20) {
  console.log("The age is between 10 and 20.");
} else {
  console.log("The age is not between 10 and 20.");
}
```

☑ Q2: Switch Statement Example

```
let day = 3;
let dayName;

switch (day) {
  case 1: dayName = "Monday"; break;
  case 2: dayName = "Tuesday"; break;
  case 3: dayName = "Wednesday"; break;
  case 4: dayName = "Thursday"; break;
  case 5: dayName = "Friday"; break;
  case 6: dayName = "Saturday"; break;
  case 7: dayName = "Sunday"; break;
  default: dayName = "Invalid day";
}

console.log(dayName); // Wednesday
```

☑ Q3: Check if number divisible by both 2 & 3

```
let number = 12;
if (number % 2 === 0 && number % 3 === 0) {
  console.log(number + " is divisible by both 2 and 3.");
} else {
  console.log(number + " is not divisible by both 2 and 3.");
}
```

🧠 Bitwise Operators in JavaScript ⚙️

Bitwise operators perform operations on binary representations of numbers. They're a powerful tool for low-level programming tasks like bit masking, flags, and optimization tricks. JavaScript treats numbers as **32-bit signed integers** during bitwise operations. Let's break down each operator:

🔸 Bitwise AND (`&`)

✅ Returns `1` **only if both bits are 1**, else returns `0`.

```
5 & 3 → 101 & 011 = 001 → 1
```

🧠 **Use case**: Masking bits, filtering flags.

🔸 Bitwise OR (`|`)

✅ Returns `1` **if at least one bit is 1**, else `0`.

```
5 | 3 → 101 | 011 = 111 → 7
```

🧠 **Use case**: Setting flags.

⚡ Bitwise XOR (`^`)

✅ Returns `1` **only if the bits are different**, else `0`.

```
5 ^ 3 → 101 ^ 011 = 110 → 6
```

🧠 **Use case**: Swapping values without a temp variable.

🔁 Bitwise NOT (`~`)

✅ **Flips** every bit — 1 becomes 0, and 0 becomes 1.

```
~5 → ~00000000 00000000 00000000 00000101 = 11111111 11111111 11111111 11111010 →
-6
```

🧠 **Note**: Result is `-(n + 1)` due to two's complement.

⬅️ Left Shift (`<<`)

✅ Shifts bits **to the left** by `n` positions. 📈 Equivalent to: `number × 2^n`

```
5 << 1 → 101 << 1 = 1010 → 10
```

🧠 **Use case**: Fast multiplication by powers of 2.

➡️ Right Shift (`>>`)

✅ Shifts bits **to the right**, preserving the sign bit (for negatives). 🧮 Equivalent to: `Math.floor(number / 2^n)`

```
5 >> 1 → 101 >> 1 = 10 → 2
```

🧠 **Use case**: Fast division by powers of 2.

⏩ Zero-Fill Right Shift (`>>>`)

✅ Similar to `>>`, **but always fills leftmost bits with 0.** 🔍 Ignores sign bit (treats number as unsigned).

```
-5 >>> 1 → large positive number (e.g. 2147483645)
```

🧠 **Use case**: Working with unsigned 32-bit values.

🧰 Quick Example

```
let a = 5;        // 0101
let b = 3;        // 0011

console.log(a & b);  // 1
console.log(a | b);  // 7
console.log(a ^ b);  // 6
console.log(~a);     // -6
console.log(a << 1); // 10
console.log(a >> 1); // 2
console.log(a >>> 1);// 2
```

⚠️ Key Notes

- 💡 JavaScript converts numbers to **32-bit signed integers** for bitwise operations.
- 🧮 Avoid using bitwise operators on very large numbers (they may behave unexpectedly).
- ⚙️ Great for **performance optimization**, **binary flags**, or **hardware-level operations**.
- 📦 Use with caution in high-level apps due to their complexity and potential confusion.

📌 Summary Table

Operator	Symbol	Meaning	Use Case	
AND	&	Both bits 1 → 1	Masking, filtering bits	
OR	`	`	At least one bit 1 → 1	Setting flags
XOR	^	Different bits → 1	Toggle/swapping bits	
NOT	~	Flips bits	Negation	
Left Shift	<<	Shift left (× 2^n)	Fast multiply	
Right Shift	>>	Shift right (÷ 2^n)	Fast divide, signed numbers	
Zero-fill Right Shift	>>>	Unsigned right shift	Fast unsigned divide	