Bitwise operators

Absolutely! Let's go in-depth on **Bitwise Operators** in JavaScript — an advanced but powerful topic that manipulates data at the **bit level** (0s and 1s).

⚙️ What Are Bitwise Operators?

Bitwise operators treat operands as **32-bit binary numbers** and operate on each **bit**.

For example:

```
5 (decimal) = 0101 (binary)
3 (decimal) = 0011 (binary)
```

🧾 Bitwise Operators Table

Operator	Name	Description	Example	Result
`&`	AND	Sets each bit to 1 if **both** bits are 1	`5 & 3`	`1`
`	`	`	OR	Sets each bit to 1 if **one** of two bits is 1	`5
`^`	XOR	Sets each bit to 1 if **only one** bit is 1	`5 ^ 3`	`6`
`~`	NOT	Inverts all the bits	`~5`	`-6`
`<<`	Left Shift	Shifts bits to the left and fills with 0s	`5 << 1`	`10`
`>>`	Right Shift	Shifts bits to the right	`5 >> 1`	`2`

`>>>`	Zero-fill Right Shift	Same as `>>` but fills left bits with 0	`-5 >>> 1`	Big Num

🔍 Breakdown of Each Operator

`&` Bitwise AND

- Only returns 1 if **both bits are 1**

```
5 & 3   // 0101 & 0011 = 0001 → 1
```

`|` Bitwise OR

- Returns 1 if **either bit is 1**

```
5 | 3   // 0101 | 0011 = 0111 → 7
```

`^` Bitwise XOR (Exclusive OR)

- Returns 1 if **only one** of the bits is 1

```
5 ^ 3   // 0101 ^ 0011 = 0110 → 6
```

`~` Bitwise NOT

- Inverts each bit (1 → 0, 0 → 1)

```
~5     // ~0101 = 1010 (in 2's complement: -6)
```

`<<` Left Shift

- Shifts all bits to the left, adds 0 on the right
- Equivalent to multiplying by 2

```
5 << 1  // 0101 << 1 = 1010 → 10
```

`>>` Right Shift

- Shifts all bits to the right, keeps the sign bit (for negatives)

```
5 >> 1  // 0101 >> 1 = 0010 → 2
```

`>>>` Zero-fill Right Shift

- Like `>>`, but fills left with **0** regardless of sign

```
-5 >>> 1  // Converts to a big positive number
```

🧠 Common Use Cases

- Low-level programming
- Performance optimization
- Working with binary flags
- Graphics and hardware programming
- Hashing and encryption algorithms

🧪 Example in JS

```
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
```

Let me know if you’d like a visual diagram of how bits shift or a quiz to test your understanding!