Bit Shifting

A **bit shift** moves each digit in a set of bits left or right. The last bit in the direction of the shift is lost, and a 0 bit is inserted on the other end.

```java
0010 << 1 // 0100
1011 >> 1 // 0101
```

Bit shifts take number of times to shift as the right argument:

```java
1010110 << 2 // 1011000
1011010 >> 3 // 0001011
```

A single left shift multiplies a binary number by 2:

```java
// 0010 is 2
0010 << 1 // 0100, which is 4
```

Two left shifts multiplies by 4. Three left shifts multiplies by 8.

And similarly, shifting right *divides* by 2, throwing out any remainders.

See also:

- Binary Numbers (/concept/binary-numbers)
- Bitwise AND (/concept/and)
- Bitwise OR (/concept/or)
- Bitwise NOT (/concept/not)
- Bitwise XOR (eXclusive OR) (/concept/xor)

What's next?

If you're ready to start applying these concepts to some problems, check out our mock coding interview questions (/next).

They mimic a real interview by offering hints when you're stuck or you're missing an optimization.

Try some questions now ➜

Want more coding interview help?

Check out **interviewcake.com** for more advice, guides, and practice questions.