

Язык JavaScript (дополнительный материал)

A large, bold, black 'JS' logo is centered on the page. The 'J' and 'S' are stylized with thick strokes and rounded terminals. The 'J' has a curved bottom, and the 'S' is a simple, thick, rounded letter.

ПОБИТОВЫЕ ОПЕРАТОРЫ

Побитовые операторы рассматривают аргументы как 32-разрядные целые числа и работают на уровне их внутреннего двоичного представления.

Поддерживаются следующие побитовые операторы:

- AND(и) ($\&$)
- OR(или) ($|$)
- XOR(побитовое исключающее или) (\wedge)
- NOT(не) (\sim)
- LEFT SHIFT(левый сдвиг) (\ll)
- RIGHT SHIFT(правый сдвиг) (\gg)
- ZERO-FILL RIGHT SHIFT(правый сдвиг с заполнением нулями) (\ggg)

ПОБИТОВЫЕ ОПЕРАТОРЫ

Логические операции **И**, **ИЛИ**, **исключающее ИЛИ** и **НЕ** могут быть описаны с помощью таблиц истинности:

Логический оператор И

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

*Логический оператор
исключающее ИЛИ*

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Логический оператор ИЛИ

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

*Логический
оператор НЕ*

X	NOT X
0	1
1	0

ПОБИТОВЫЕ ОПЕРАТОРЫ

В побитовых операциях значение бита, равное 1, рассматривается как логическая истина, а 0 как ложь. Побитовое И (оператор &) берёт два числа и логически умножает соответствующие биты. Например:

```
let a = 3;           000000011
let b = 8;           00001000
let c = a & b;        ↓↓ ↓↓ ↓↓ ↓↓ ↓↓ ↓↓ ↓↓
console.log(c); // 0  000000000
```

```
let a = 31;          00011111
let b = 17;           00010001
let c = a & b;        ↓↓ ↓↓ ↓↓ ↓↓ ↓↓ ↓↓ ↓↓
console.log(c); // 17 00010001
```

Логический оператор И

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

ПОБИТОВЫЕ ОПЕРАТОРЫ

Аналогично работает операция побитового ИЛИ (оператор `|`), за исключением того, что она логически суммирует соответствующие биты чисел без переноса. Например:

```
let a = 15;           00001111
let b = 11;           00001011
let c = a | b;         ↓↓↓↓↓↓↓↓
console.log(c); // 15  00001111
```

```
let a = 33;           00100001
let b = 11;           00001011
let c = a | b;         ↓↓↓↓↓↓↓↓
console.log(c); // 43  00101011
```

Логический оператор ИЛИ

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

ПОБИТОВЫЕ ОПЕРАТОРЫ

Побитовое отрицание (оператор `~`) работает не для отдельного бита, а для всего числа целиком. Оператор инверсии меняет ложь на истину, а истину на ложь, для каждого бита. Например:

```
let a = 65;           01000001
let b = ~a;           ↓↓ ↓↓ ↓↓ ↓↓
console.log(b); // -66 10111110
```

Логический
оператор НЕ

X	NOT X
0	1
1	0

ПОБИТОВЫЕ ОПЕРАТОРЫ

Исключающее ИЛИ (оператор \wedge) применяет побитовую операцию XOR. Например:

```
let a = 12;           00001100,
let b = 85;           01010101
let c = a ^ b;         ↓↓ ↓↓ ↓↓ ↓↓ ↓↓ ↓↓ ↓↓
console.log(c); // 89  01011001
```

*Логический оператор
исключающее ИЛИ*

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

!!! Обратимая операция.

ПОБИТОВЫЕ ОПЕРАТОРЫ

Операций сдвига: **битовый сдвиг влево** (оператор \ll) и **битовый сдвиг вправо** (оператор \gg). Битовый сдвиг вправо сдвигает биты числа вправо, слева добавляется *копия* крайнего-левого бита. Битовый сдвиг влево: сдвигает биты влево, дописывая справа нули. Вышедшие за пределы числа биты отбрасываются. Например, сдвиг числа 5 влево на 2 позиции

$$00000101 \ll 2 == 00010100$$

Сдвиг числа 19 вправо на 3 позиции

$$00010011 \gg 3 == 00000010$$

Так как сдвиг вправо (\gg) дописывает слева нули, то для целых чисел операция равносильна целочисленному делению пополам, а сдвиг влево умножению на 2.

ЗАДАЧА

Шифрование строки по ключу.

Пользователь вводит произвольную строку.
Затем вводит ключ (один символ).

Затем программа шифрует текст, а именно применяет операцию исключающего ИЛИ для каждого символа введённой строки с использованием ключа.

Вывести зашифрованное сообщение.

ЧТО ПОЧИТАТЬ ДОПОЛНИТЕЛЬНО

<https://learn.javascript.ru/bitwise-operators> - Побитовые операторы