

Побітові оператори

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators#bitwise_operators

<https://uk.javascript.info/operators>

https://w3schoolsua.github.io/js/js_bitwise.html#gsc.tab=0

Побітові оператори

- Побітові оператори розглядають аргументи як 32-бітні цілі числа та працюють на рівні їхнього двійкового представлення.
- Ці оператори не є специфічними для JavaScript. Вони підтримуються у більшості мов програмування.
- Список операторів:
 - AND(i) (&)
 - OR(або) (|)
 - XOR(побітове виключне або) (^)
 - NOT(ні) (~)
 - LEFT SHIFT(зсув ліворуч) (<<)
 - RIGHT SHIFT(зсув праворуч) (>>)
 - ZERO-FILL RIGHT SHIFT(зсув праворуч із заповненням нулями) (>>>)

Оператор	Ім'я	Опис
&	AND	Встановлює кожен біт в 1, якщо обидва біти 1
	OR	Встановлює кожен біт в 1, якщо один із двох бітів 1
^	XOR	Встановлює кожен біт в 1, якщо лише один із двох бітів 1
~	NOT	Інвертує всі біти
<<	Здвиг ліворуч з нульовим заповненням	Здвигає ліворуч, вставляючи нулі праворуч та дозволяючи крайнім лівим бітам відпасти
>>	Здвиг праворуч	Здвигає праворуч, вставляючи копії крайнього лівого біта зліва і дозволяючи крайнім правим бітам відпасти
>>>	Здвиг праворуч з нульовим заповненням	Здвигає праворуч, вводячи нулі зліва і дозволяючи крайнім правим бітам відпасти

Оператор	Ім'я	Опис
&	AND	Встановлює кожен біт в 1, якщо обидва біти 1
	OR	Встановлює кожен біт в 1, якщо один із двох бітів 1
^	XOR	Встановлює кожен біт в 1, якщо лише один із двох бітів 1
~	NOT	Інвертує всі біти
<<	Здвиг ліворуч з нульовим заповненням	Здвигає ліворуч, вставляючи нулі праворуч та дозволяючи крайнім лівим бітам відпасти
>>	Здвиг праворуч	Здвигає праворуч, вставляючи копії крайнього лівого біта зліва і дозволяючи крайнім правим бітам відпасти
>>>	Здвиг праворуч з нульовим заповненням	Здвигає праворуч, вводячи нулі зліва і дозволяючи крайнім правим бітам відпасти

Побітове AND

Коли побітове AND виконується для пари бітів, він повертає 1, якщо обидва біти дорівнюють 1.

Однобітовий приклад:

Операція	Результат
0 & 0	0
0 & 1	0
1 & 0	0
1 & 1	1

4-бітовий приклад:

Операція	Результат
1111 & 0000	0000
1111 & 0001	0001
1111 & 0010	0010
1111 & 0100	0100

Побітове OR

Коли для пари бітів виконується побітове OR, воно повертає 1, якщо один із бітів дорівнює 1:

Однобітовий приклад:

Операція	Результат
0 0	0
0 1	1
1 0	1
1 1	1

4-бітовий приклад:

Операція	Результат
1111 0000	1111
1111 0001	1111
1111 0010	1111
1111 0100	1111

Побітове XOR

Коли побітове XOR виконується для пари бітів, повертає 1, якщо біти різні:

Однобітовий приклад:

Операція	Результат
0 ^ 0	0
0 ^ 1	1
1 ^ 0	1
1 ^ 1	0

4-бітовий приклад:

Операція	Результат
1111 ^ 0000	1111
1111 ^ 0001	1110
1111 ^ 0010	1101
1111 ^ 0100	1011

JavaScript Побітове AND (&)

Побітове AND повертає 1 тільки якщо обидва біти дорівнюють 1:

[illegible]

JavaScript Побітове OR (|)

Побітове OR повертає 1, якщо один із бітів дорівнює 1:

Десяткове	Бінарне
5	00000000000000000000000000000101
1	00000000000000000000000000000001
5 1	00000000000000000000000000000101 (5)

JavaScript Побітове XOR (^)

Побітове XOR повертає 1, якщо біти різні:

[illegible]

Можна використати для збереження окремих прав користувача. При цьому для збереження кожного із прав використовуємо тільки один біт.

Приклад.

	Видаляти	Додавати	Змінювати	Читати
.....	0	0	1	1

```
const READ_ACCESS = 1    // 0      0      0      1
const CHANGE_ACCESS = 2  // 0      0      1      0
const ADD_ACCESS = 4     // 0      1      0      0
const DELETE_ACCESS = 8  // 1      0      0      0
```

Приклад.

	Видаляти	Додавати	Змінювати	Читати
.....	0	0	1	1

Значення бітів як числа (`parseInt('0011',2) = 3`)

Встановлення прав

let permissions = 3 $//3 = 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

.....	2^3	2^2	2^1	2^0
.....	8	4	2	1
.....	0	0	1	1

Задати правила

Приклад.

	Видаляти	Додавати	Змінювати	Читати
.....	0	0	1	1

1

CHANGE_ACCESS = 2 // 0 0 1 0

.....

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \end{array}$$

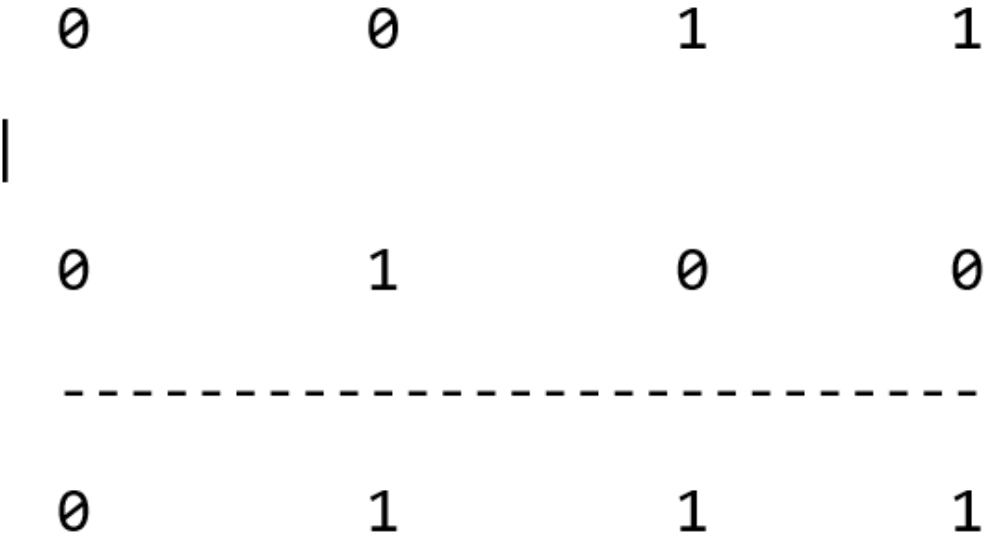
Додати правило (якщо не було – додамо, якщо було - залишиться)

```
let permissions = permissions | нове_правило
```

Приклад

```
permissions = permissions | ADD_ACCESS
```

const READ_ACCESS = 1	//	0	0	0	1
const CHANGE_ACCESS = 2	//	0	0	1	0
const ADD_ACCESS = 4	//	0	1	0	0
const DELETE_ACCESS = 8	//	1	0	0	0



	Видаляти	Додавати	Змінювати	Читати
.....	0	1	1	1

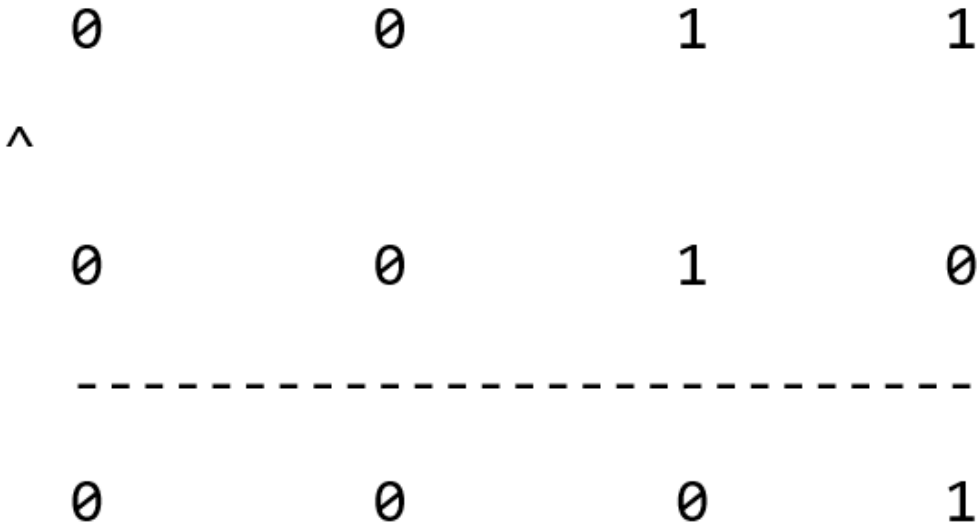
Змінити правило на протилежний стан (не було – встановити, було - забрати)

```
let permissions = permissions ^ правило_для_зміни
```

Приклад

```
permissions = permissions ^ CHANGE_ACCESS
```

const READ_ACCESS = 1	//	0	0	0	1
const CHANGE_ACCESS = 2	//	0	0	1	0
const ADD_ACCESS = 4	//	0	1	0	0
const DELETE ACCESS = 8	//	1	0	0	0



	Видаляти	Додавати	Змінювати	Читати
.....	0	0	0	1

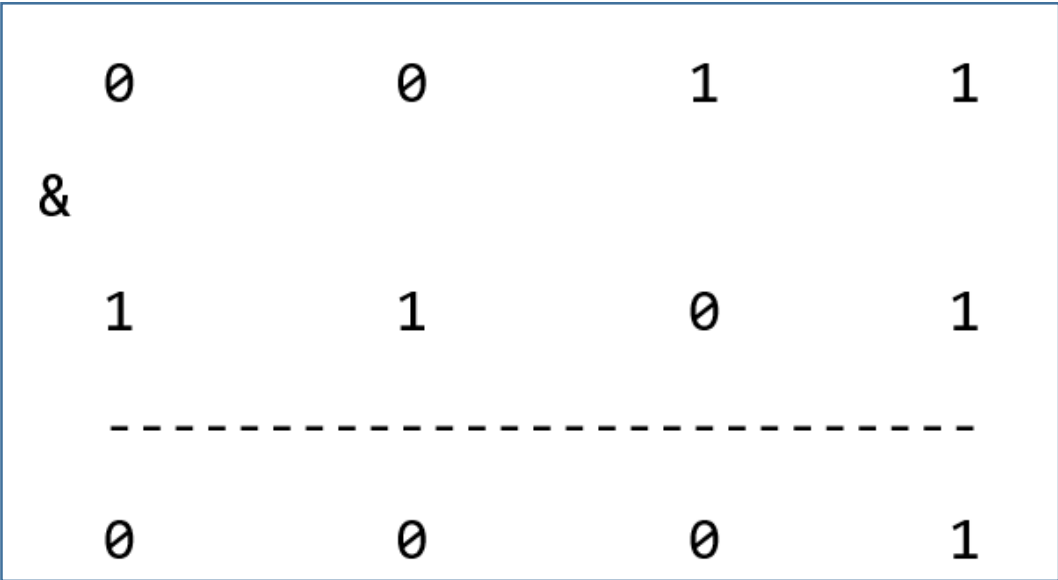
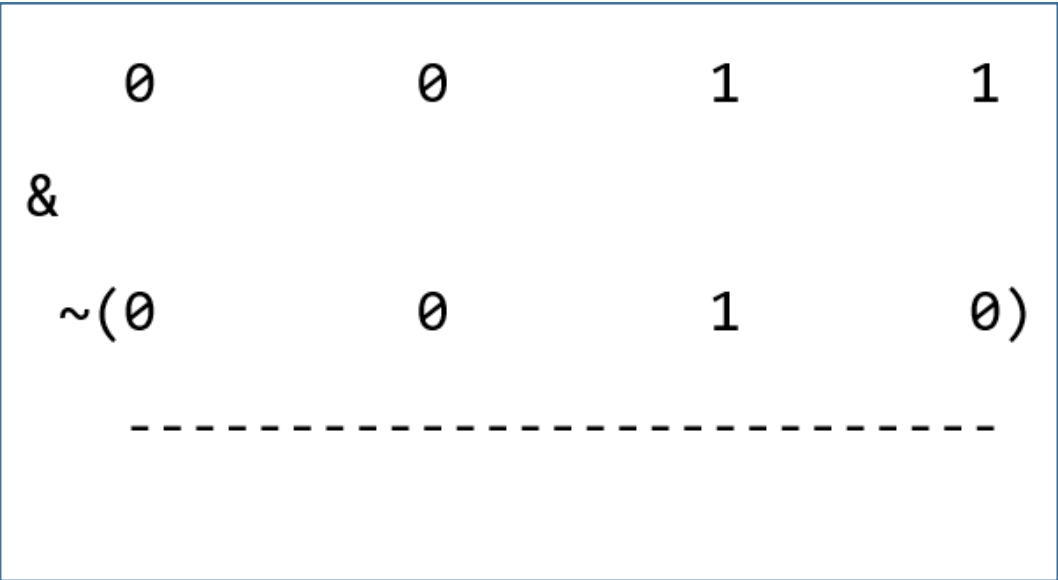
Гарантовано забрати правило

let permissions = permissions & **~правило_для_гарантованої_відмови**

Приклад

permissions = permissions & **~CHANGE_ACCESS**

const READ_ACCESS = 1	//	0	0	0	1
const CHANGE_ACCESS = 2	//	0	0	1	0
const ADD_ACCESS = 4	//	0	1	0	0
const DELETE_ACCESS = 8	//	1	0	0	0



	Видаляти	Додавати	Змінювати	Читати
.....	0	0	0	1

Перевірити чи має право

permissions & *правило_для_перевірки*

Приклад

permissions & **CHANGE_ACCESS**

0 0 1 1

&

0 0 1 0

0 0 1 0

const READ_ACCESS = 1	//	0	0	0	1
const CHANGE_ACCESS = 2	//	0	0	1	0
const ADD_ACCESS = 4	//	0	1	0	0
const DELETE_ACCESS = 8	//	1	0	0	0

	Видаляти	Додавати	Змінювати	Читати
.....	0	0	1	1

```
if (permissions & CHANGE_ACCESS)  
  console.log('Можна змінювати')  
else  
  console.log('Не можна змінювати')
```