**Використання декількох умовних (трикомпонентних) операторів**

У попередньому завданні ви використовували одинарний умовний оператор. Ви також можете їх об'єднати для того, щоб перевірити декілька умов.

Наступна функція використовує if, else if, а також else твердження, щоб перевірити декілька умов:

function findGreaterOrEqual(a, b) {

if (a === b) {

return "a and b are equal";

}

else if (a > b) {

return "a is greater";

}

else {

return "b is greater";

}

}

Вказану вище функцію можна переписати, використовуючи декілька умовних операторів:

function findGreaterOrEqual(a, b) {

return (a === b) ? "a and b are equal"

: (a > b) ? "a is greater"

: "b is greater";

}

Найкращою практикою вважають відформатування декількох умовних операторів таким чином, що кожна умова буде знаходитися в окремому рядку, як вказано вище. Використання декількох умовних операторів без правильних відступів може ускладнити читання вашого коду. Наприклад:

function findGreaterOrEqual(a, b) {

return (a === b) ? "a and b are equal" : (a > b) ? "a is greater" : "b is greater";

}

У функції checkSign, використовуйте декілька умовних операторів, дотримуючись рекомендованого формату, що використаний у findGreaterOrEqual - для того, щоб перевірити чи число додатнє, від'ємне або нуль. Функція має повернути positive, negative or zero.

Запустити тест (Ctrl + Enter)Скинути весь код

Отримати допомогу

**Tests**

* В очікуванні:checkSign має використовувати декілька умовних операторів
* В очікуванні:checkSign(10) має повернути рядок positive. Зверніть увагу на те, що написання з великої літери - важливе
* В очікуванні:checkSign(-12) має повернути рядок negative. Зверніть увагу на те, що написання з великої літери - важливе
* В очікуванні:checkSign(0) має повернути рядок zero. Зверніть увагу на те, що написання з великої літери - важливе

function checkSign(num) {

return (num == 0) ? "zero" : (num > 0) ? "positive" : "negative";

}

checkSign(10);