Logical Operators

Sometimes the information you want is how multiple boolean values relate to each other, and in that case it's convenient to be able to use logical operator to separate these boolean values.

Logical "And" Operator

The first logical operator I'm going to introduce you to is the and operator, and an and operator is represented by two ampersands next to each other. If you create an expression comparing multiple values separated by double ampersands, you'll get the value `true` only if all of the values inside of that expression are true .If there are no booleans in the comparisons, the return value will be the last `true` value. That might sound a little bit complex but let me show it to you, and I think you'll understand.

```
var truth = true;
var falsehood = false;
var firstWord = "Hello";
var secondWord = "World";

console.log(truth && truth);
// true

console.log(truth && falsehood);
// false

console.log((3 < 4) && ("Banana" > "Apple"));
// true

console.log(firstWord && secondWord);
// "World"

console.log(secondWord && firstWord);
// "Hello"

console.log(falsehood && firstWord);
// false
```

Logical "Or" Operator

There's also a logical or operator. With that operator multiple values that are separated by double pipes will return `true` if any of those values is `true`. If there are no other true values, the first `true` value will be returned.

```
var truth = true;
var falsehood = false;
var firstWord = "Hello";
var secondWord = "World";

console.log(falsehood || falsehood);
// false

console.log(truth || falsehood);
```

```
// true

console.log((3 < 4) || ("Banana" > "Apple"));
// true

console.log(firstWord || secondWord);
// "Hello"

console.log(secondWord || firstWord);
// "World"

console.log(falsehood || firstWord);
// "Hello"
```

Logical "Not" Operator

You can also use a logical not in order to invert the value of an expression being compared. A single exclamation mark will negate the boolean value of a variable or an expression.

```
var truth = true;
var falsehood = false;

console.log(!truth);
// false

console.log(!falsehood);
// true

console.log(!(3 < 4));
// false

console.log(!(3 < 4) || ("Banana" > "Apple"));
// true
```

Logical "Not" for Primitive Variables

The boolean value of any non-empty and non-zero string or any number other than zero with not is considered `false`. The boolean value of an empty string, a zero number, or a `null` or `undefined` variable with a logical not is considered `true`.

```
var word = "Hello";
var empty = "";
var negative = -10;
var zero = 0;
var nullValue = null;
var notDefined;

console.log(!word);
// false

console.log(!empty);
// true

console.log(!negative);
// false

console.log(!zero);
// true

console.log(!nullValue);
// true

console.log(!notDefined);
// true
```

Deriving a Boolean Value

You can also derive the boolean value of a variables imply by using two exclamation marks. Essentially, you're doing not-not. That'll convert any primitive value to its boolean equivalent. The boolean value of non-empty and non-zero strings or numbers is `true`, and the boolean value of an empty string, or zero value, or a no or `undefined` variable is `false`, and I can demonstrate that as well.

```
var word = "Hello";
var empty = "";
var negative = -10;
var zero = 0;
var nullValue = null;
var notDefined;

console.log(!!word);
// true

console.log(!!empty);
// false

console.log(!!negative);
// true

console.log(!!zero);
// false

console.log(!!nullValue);
// false

console.log(!!notDefined);
// false
```

This trick with the double exclamation marks is convenient to know about when you need to figure out the boolean value of a given variable or value.

Logic Precedence

When you're using the logical comparison operators, it's important to note the precedence in which they're executed. Basically all you need to know is that evaluation for a series of logical operators within a single expression, is always done from left to right.

```
var truth = true;
var falsehood = false;

console.log(truth && falsehood && truth);
// false

console.log(truth || falsehood || truth);
// true

console.log(truth && falsehood || truth);
// true

console.log(truth || falsehood && truth);
// true

console.log(truth && falsehood || falsehood);
// false
```

Get comfortable with some of these logic operators and figuring out the precedence of the calculations. When you experiment with them a little bit you're going to get familiarwith how they work.