Operators & Arithmetic

In order to look at implicit type coercion in the next video, we'll be using some operators, so I thought that this would be a good time to go over JavaScript operators. Most of these are in just about every language. This is really easy stuff, but I want to make sure we cover everything.

We have a few different types of operators

Arithmetic Operators

Addition, subtraction, multiplication and division

```
5 + 5; // 10
```

```
10 - 5; // 5
```

```
5 * 5; // 25
```

```
10 / 2; // 5
```

Concatenation: The + operator can also be used to put 2 or more strings together. The use of + depends on the data type

```
'Hello, ' + 'World!'; // Hello World!
```

Modulus is used to get a division remainder

```
10 % 3; // 1
```

Exponentiation returns the result of the first operand to the power of the second.

```
2**3; // 8
```

Increment is used to **increment** or add 1 to a value

```
x = 10;
x++; // 11
```

Decrement is used to **decrement** or remove 1 from a value

```
x = 10;
x--; // 9
```

Assignment Operators

Assignment operators do something with the value on the right side to set the variable on the left side.

Value assignment

```
x = 10;
```

Addition assignment operator adds the amount on the right side

```
x = 10;
x += 10; // 20
// Same as x = x + 10
```

Subtraction assignment operator subtracts the amount on the right side

```
x = 10;
x -= 10;  // 0
// Same as x = x - 10
```

Multiplication assignment operator multiplies the amount on the right side

```
x = 10;
x *= 10; // 100
// Same as x = x \* 10
```

Division assignment operator divides the amount on the right side

```
x = 10;
x /= 10; // 1
// Same as x = x / 10
```

Modulo assignment operator gets division remainder of the amount on the right side

```
x = 10;
x %= 10; // 0
// Same as x = x % 10
```

Exponentiation assignment operator gets exponent of the amount on the right side

```
x = 10;
x **= 10; // 100
// Same as x = x \*\* 10
```

Comparison Operators

Comparison operators are used to compare values

Equal to operator - The following will return true even if the types do not match

```
2 == 2 // true
2 == '2' // true
```

Equal value & equal type - The types have to match or it will return false

```
2 === 2 // true
2 === '2' // false
```

Not equal to operator - returns true if not equal

```
2 != 2 // false
2 != '2' // false
```

Not equal value & equal type

```
2 !== 2 // false
2 !== '2' // true
```

Greater than

```
10 > 5 // true
```

Less than

```
10 < 5 // false
```

Greater than or equal

```
10 >= 5 // true
```

Less than or equal

```
10 <= 5 // false
```

== vs ===

As I said above, the `==` operator is used to compare values. The `===` operator is used to compare values and types.

The double equals (==) performs type coercion, which means it makes sure that the values are the same type before comparing them.

Which one you use is up to you, but I personally use the triple equals (===) because it is more explicit and your code will be less prone to errors. In many situations, it doesn't matter, but I prefer to use it unless there is a specific reason not to.

Later on we will talk about **truthy** and **falsy** values and I will show you some situations where the double equals can cause issues.