Numbers and Math in ES6

WE'LL COVER THE FOLLOWING

In ES6 there are a bunch of additions to Numbers and the Math object. There are new Number literals, Binary and Octal, as well a ton of new methods for `Math`.

Binary Literals

In ES6 we have to ability to create binary literals. In order to do this we need to prefix our binary number with `0b` or `0B`.

```
console.log(0b001) // The number 1
console.log(0b010) // The number 2
console.log(0b011) // The number 3
console.log(0b100) // The number 4
```

If you have never worked with binary numbers before, the pattern goes like this: Starting for the right to the left, the smallest number is as far right as possible. For example `0001` in binary is `1`.

8	4	2	1
0	0	0	1

If we wanted to get the numbers `3` it would be `0011`, adding `2` and `1`.

8	4	2	1
0	0	1	1

Let's do one more, how about `10`? Well that would be `1010` or `8` + `2`.

8	4	2	1
1	0	1	0

Math with binary numbers

We can perform mathematical operations with the binary numbers just like we normally would.

```
console.log(0b100 + 0b010)
```

Can you guess what that adds up too? Well `6` of course, the neat thing is we can add binary and integer numbers together.

```
console.log(0b110 + 9)
```

This will produce `15`.

Octal Literals

Just like binary literals we are able to create octal literals as well. Octal numbers are a base 8 number system, binary is a base 2 system. In Octal numbers each place is a multiple of 8, and the number present in that place will be used to determine the value. Let's look at some examples

512	64	8	1
0	0	0	0

Like the binary literals we use `0o` or `00` to denote it what number type it will be, so `0o0001` will be the number `1`.

512	64	8	1
0	0	0	1

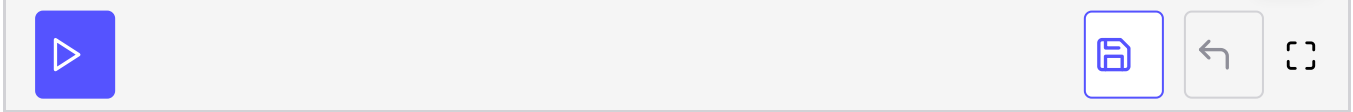
If we had `0o0011` we would get `9`. Now we don't just have to put a `0` or `1` in the place here, we can use a number from `0` to `7`.

So if we had `0o0020` this would be `16`, as it takes the multiple in that place and times it by `2`.

Math with octal numbers

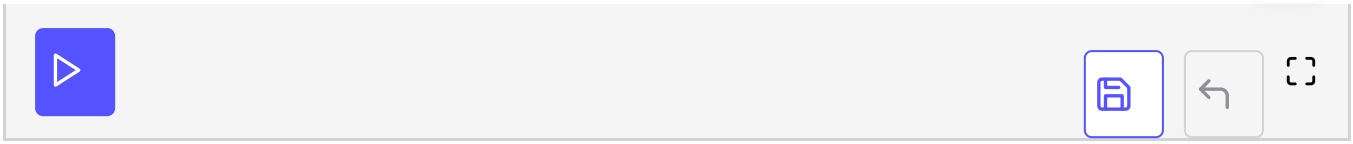
Just like the binary literal, octals can be used to preform mathematical operations on with mixed number types.

```
console.log(0o0020 + 7) // 23
```



We can even mix it with binary numbers.

```
console.log(0o0010 + 0b0111) // 15
```



Math

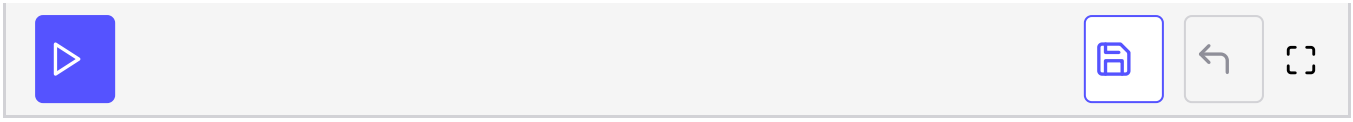
In ES6 there are a ton of new Math methods. These include `Math.clz32`, `Math.imul`, `Math.sign`, `Math.log10`, `Math.log2`, `Math.log1p`, `Math.expm1`, `Math.cosh`, `Math.sinh`, `Math.tanh`, `Math.acosh`, `Math.asinh`, `Math.atanh`, `Math.trunc`, `Math.fround`, `Math.cbrt`, `Math.hypot`...woof that is a lot! A lot of the additions here are for Trigonometry.

We won't look at all of these, but I would like to show you one method, that is `Math.trunc`.

Math.trunc

The `.trunc()` method is used to truncate the number provided to it, for example:

```
console.log(Math.trunc(5.6) )//5
console.log(Math.trunc(9.231)) //9
```



It is used to just cut off the numbers after the decimal, this differs from `Math.floor`, `Math.ceil` and `Math.round` in that it just cuts them off, it does not perform any rounding on the number.