Number Extensions

in-built methods in ES6 to play with numbers

Finally, ES6 gives you a built-in <code>isInteger</code> check. Earlier, we had to implement different tricks to determine if a number is an integer. These tricks were not too semantic. In ES6, all we need to do is call



Safe integers are integers that can be represented in JavaScript without loss of precision.

```
let maxVal = Number.MAX_SAFE_INTEGER;
let minVal = Number.MIN_SAFE_INTEGER;

console.log(Number.isSafeInteger( maxVal ));
//> true

console.log(Number.isSafeInteger( maxVal + 1 ));
//> false
```

Number. EPSILON represents the difference between one and the smallest value greater than one. We can use this constant to compare values against each other considering that arithmetic operations may be distorted. Example:

```
console.log( 0.1 + 0.2 <= 0.3 );
//> false

console.log( 0.1 + 0.2 <= 0.3 + Number.EPSILON );
//> true
```

The parseInt and parseFloat methods are now available from the Number object. In order to provide more context, I recommend using Number.parseInt and Number.parseFloat from now on.

The global isFinite and isNaN functions are also available in the Number object.

There is one difference between the old global functions and the new Number methods:

```
console.log( isFinite( '0' ), Number.isFinite( '0' ) );
//> true false

console.log(
   isNaN( 'IamNotANumber' ),
   Number.isNaN( 'IamNotANumber' ) );
//> true false
```

Number methods keep the types.

In the next chapter, we will study some features that Es2016 provides us and how they make writing javascript code easier.