**ECMAScript 2015 Goodies – Part 3 – binary and octal number literals**

This is the third article in ECMAScript 2015 Goodies. In the earlier articles we covered new keywords in the language – let and const, which allows you to define a true block level variable and a constant respectively in the program. In this article, we will cover two additional number literals introduced in the language – binary and octal. These new number literals helps you to easily identify the different types of numbers [binary, octal, hex, decimal] as they are prefix with specific value.

The primary reason for adding these two literals in the language was a decimal literal in JavaScript can start with number zero followed by another decimal digit. However if the number after leading zero is less than 8, that number gets parsed as octal number, rather than decimal. This has caused a lot of confusion and bugs amongst developer community, so the new number literals will help you to clearly define different types of literals and also makes your program more readable.

Let’s understand the syntax and rules while defining these literals.

**Binary**

Binary number syntax uses a leading zero followed by a lowercase or uppercase Latin letter "B" (0b or 0B). The digits after 0b or 0B must be either 0 or 1, or else JavaScript compiler will throw a syntax error.

var validBinary = 0b1111;

var invalidBinary = 0BX1111; //SyntaxError

console.log(validBinary); //15

**Octal**

Octal number syntax uses a leading zero followed by a lowercase or uppercase Latin letter "O" (0o or 0O). The digits after 0o or 0O must be in the valid range from 0 to 7, else JavaScript compiler will throw a syntax error.

var validOctal = 0o1234;

var invalidOctal = 0OAA1234; // SyntaxError

console.log(validOctal); //668

That’s all I wanted to cover in this article. Thanks for reading.