Episode 7.06 - Stupid Binary Tricks

(Transcript URL: https://intermation.com/episode-7-06-stupid-binary-tricks/)

Show Description: Having learned how to program bitwise operations, it is now time to flex our bit bashing muscles by investigating some creative ways to perform common programming functions.

Try it Yourself

All of the code presented in this worksheet can be executed in a JavaScript-enabled browser. No compiler or other software development tool is needed. There are two ways to do this:

- Copy the code into a text editor such as Notepad (Windows) or TextEdit (Mac), and save the file with the extension **`.htm`**. Locate the file on your computer and open it in a browser. Some tablets and smartphones allow you to store a text file to the file system and open it in a browser, but the process is more complicated.
- Alternatively, you can use a web-based tutorial service such as https://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst. Replace the code in their editor window with the code shown below, and then run it.

Podcast Timestamp	Supporting Details

4:29

Using Bitwise Operations to Determine the Least Significant One of an Integer

The code below finds the least significant one in an integer, which is the same as finding the largest power of two that is a divisor of the integer.

```
<!DOCTYPE html>
<html>
<head>
    <title>Determining the Least Significant One in an Integer
           - intermation.com</title>
</head>
<body>
    <p>
    <script>
        var x = 24;
        document.write("The least significant one in " + x);
        document.write(" is at bit position " + (-x & x));
    </script>
    </p>
</body>
</html>
```

Expected Output

The least significant one in 24 is at bit position 8

Podcast Timestamp	Supporting Details

Using Bitwise Operations to Find the Most Significant One of a Positive Integer

6:05

```
<!DOCTYPE html>
<html>
<head>
    <title>Finding Most Significant One in a positive Integer
           - intermation.com</title>
</head>
<body>
    <p>
    <script>
        var x = 4853;

// Duplicate each one in integer to position to its right.
        MS_one = x | (x >> 1);
// Duplicate the pairs of ones to the positions to the right.
        MS_one |= MS_one >> 2;
// Duplicate groups of four ones to the positions to the
right.
        MS_one |= MS_one >> 4;
// Duplicate groups of 8 ones to the positions to the right.
        MS_one |= MS_one >> 8;
// Last, duplicate groups of 16 ones to positions to the
right.
        MS_one |= MS_one >> 16;

// Now, the most significant has all zeros to its left. Adding
// one to the modified value of x turns all of the ones to
zero // and places carry in position immediately to left of MS
one.
        MS_one++;

// Shift this value one position to the right, and we have our
// most significant one.
        MS_one >>>= 1;
        document.write("The most significant one in " + x);
        document.write(" is at bit position " + MS_one);
    </script>
    </p>
</body>
</html>
```

Expected Output

```
The most significant one in 4853 is at bit position 4096
```

Podcast Timestamp	Supporting Details
11:09	***Using Bitwise Operations to Determine the Absolute Value of an Integer***

```
<!DOCTYPE html>
<html>
<head>
    <title>Determining the Absolute Value of an Integer
           - intermation.com</title>
</head>
<body>
    <p>
    <script>
        var x = -9321675;

// Creating a variable "sign" where all thirty-two bits equal
// the sign bit of x.
        var sign = x >> 31;

// If x is negative, we want to convert it using the two’s
// complement method of flipping all the bits and adding one,
// which is the same as subtracting one, then flipping all of
// the bits. A bitwise-XOR with "sign" as the mask flips all
// bits when x is negative and leaves them unflipped
otherwise.
        absoluteValueOfX = (x + sign) ^ sign;

        document.write("The absolute value of " + x);
        document.write(" is " + absoluteValueOfX);
    </script>
    </p>
</body>
</html>
```

Expected Output

```
The absolute value of -9321675 is 9321675
```

Podcast Timestamp	Supporting Details

Using Bitwise Operations to Determine if an Integer is a Power of Two

15:29

```
<!DOCTYPE html>
<html>
<head>
    <title>Determining if an Integer is a Power of Two
           - intermation.com</title>
</head>
<body>
    <p>
    <script>
        var x = 255;

// If the bitwise-AND of x with one minus x is zero, then
// x is a power of two. Two items of note. First, the most
// negative two's complement value, 1000...00 or
// ~((~0) >>> 1), will test as true.  Second, since no
// power of two is negative, we do not take the absolute
// value before determining if x is a power of two.

        if((x & (x - 1)) == 0)
            document.write(x + " is a power of two.");
        else
            document.write(x + " is not a power of two.");

    </script>
    </p>
</body>
</html>
```

Expected Output:

```
255 is not a power of two
```

Podcast Timestamp	Supporting Details
17:09	***Using Bitwise Operations to Determine if an Integer will Fit in 'n' Bits***

```
<!DOCTYPE html>
<html>
<head>
    <title>Determining if an Integer will Fit in 'n' Bits
           - intermation.com</title>
</head>
<body>
   <p>
   <script>
      var x = 15;
      var n = 4;

      var sign = x >> (n + (~0));

      if((sign ^ (sign >> 1)) == 0)
         document.write(x + " will fit in " + n + " bits");
      else
         document.write(x + " will not fit in " + n + " bits");

   </script>
   </p>
</body>
</html>
```

Expected Output:

15 will not fit in 4 bits