Episode 7.04 - Setting Bits using the Bitwise-OR

(Transcript URL: https://intermation.com/episode-7-04-setting-bits-using-the-bitwise-and/)

Show Description: The ability to set bits may not seem important, but many algorithms in computing depend on just that. Join us as we control bits and build integers from scratch using the bitwise-OR.

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.

The code shown below is presented to show how we can assign values in binary to a variable, then display them in binary. Just as “0x” added to the beginning of a number declares its base as hexadecimal, a “0b” prepended to a binary number indicates its base is base-2 to the browser. When outputting a value, appending “.toString(n)” to the variable where the ‘n’ inside the parenthesis represents the base will output the string in that base.

This snippet of code assigns 10010100 to the variable lightingControl, and then outputs it to the browser window. Note that the actual code is in **bold** and that the function **`document.write()`** outputs text to the browser window.

```
<!DOCTYPE html>
<html>
<head>
    <title>Binary Assignment and Output Example using JavaScript
          - intermation.com</title>
</head>
<body>
<script>
    var lightingControl = 0b10010100;
    document.write("Lighting Control Byte = " +
                   lightingControl.toString(2));
</script>
</body>
</html>
```

Expected Output

Lighting Control Byte = 10010100

Podcast Timestamp	Supporting Details

1:48

Using a Bitwise-OR to Control a Lighting Example

The code below defines constants representing each lighting bit position and shows how the bitwise-OR can be used to turn on individual banks of lights.

```
var lightingControl = 0;
const houseLightingMask = 128;
const workLightingMask = 64;
const aisleLightingMask = 32;
const exitLightingMask = 16;
const emergencyLightingMask = 8;
const stageLightingMask = 4;
const orchestraPitLightingMask = 2;
const curtainLightingMask = 1;

lightingControl = houseLightingMask | exitLightingMask |
stageLightingMask;

document.write("Lighting Control Byte (before) = " +
                    lightingControl.toString(2) + "<br>");

lightingControl |= (aisleLightingMask |
emergencyLightingMask);

document.write("Lighting Control Byte (after) = " +
                    lightingControl.toString(2));
```

5:11

Using Left Shifts and Bitwise-ORs to Create a 50% Gray 24-Bit RGB Value

The code below defines the color gray using 0x80 for red, green, and blue, and then uses the color to set the style color for the text output “Hello!”

```
<!DOCTYPE html>
<html>
<head>
    <title>Setting Text Color using Bitwise-OR
           - intermation.com</title>
</head>
<body>
<script>
    var fiftyPercentGray = (128 << 16) | (128 << 8) | 128;
    document.write("<span style='color:#" +
                    fiftyPercentGray.toString(16) + "'>
                    Hello!</span>");
</script>
</body>
```

Podcast Timestamp	Supporting Details
	`</html>`
6:13	***Creating a 24-Bit RGB Color from its Components*** `var color = (red << 16) | (green << 8) | blue;`

7:35	***Creating an IPv4 Address and Subnet Mask from Separate Integers***
	`address = (addressByte1 << 24) | (addressByte2 << 16) | (addressByte3 << 8) | addressByte4;`
	`subnetMask = (maskByte1 << 24) | (maskByte2 << 16) | (maskByte3 << 8) | maskByte4;`