JS-CC-009 : Convert Numbers

This code challenge has two sub tasks. You are requested to find solutions to convert binary number to decimal and decimal number to binary.

For the purpose of this code challenge, please write your own code rather than using built-in functions to convert numbers to different base systems. Here below built-in functions in JavaScript and Python:

JavaScript:

- Binary to decimal

```
var binNum = "11110";
var decNum = parseInt(binNum, 2);
console.log(decNum);
// outputs 30
```

- Decimal to binary

```
var decNum = 30;
var binNum = decNum.toString(2);
console.log(binNum);
// outputs "11110"
```

Python:

- Binary to decimal

```
binNum = '0b11110'
int(binNum, 2)
# outputs 30
```

- Decimal to binary

```
decNum = 30
bin(decNum)
# outputs '0b11110'
```

Learning Outcomes

At the end of the this coding challenge, students will be able to;

- Analyze a problem, identify and apply programming knowledge for appropriate solution.
- Demonstrate their knowledge of algorithmic design principles by using JavaScript and Python effectively.

Problem Statement

- Write a function that takes a binary number and converts it to decimal integer.
- Write a function that takes a decimal number and converts it to binary number in string representation.
- Please solve the problem for non-negative values only.

☺ **Happy Coding** ✍

Binary to decimal

JavaScript

```
function bin2dec(binStr) { // binStr is in string type

    // returning value should be in int type.
    return null;
}
```

Python

```
def bin2dec(binStr): # binStr is in str type

    # returning value should be in int type.
    return None
    pass
```

Decimal to binary

JavaScript

```
function dec2bin(decNum) { // decNum is int type

    // returning value should be in str type.
    return null;
}
```

Python

```
def dec2bin(decNum): # decNum is int type

    # returning value should be in str type.
    return None
    pass
```

Solutions

JavaScript

```javascript
function bin2dec(binStr) {
  var decVal = +0;
  var bits = +1;
  for (var i = 0; i < binStr.length; i++) {
    var currNum = +binStr[binStr.length - i - 1];
    if (currNum === 1) {
      decVal += bits;
    }
    bits *= 2;
  }
  return decVal;
}

function dec2bin(decNum) {
  if (decNum == 0) return "0";

  var binVal = "";
  while (decNum != 0) {
    binVal += +decNum % 2;
    decNum >>= 1;
  }
  return binVal.split("").reverse().join("");
}

/* *** Tests ***   */

var desc = "with zero";
var input = "0";
var actual = bin2dec(input);
var expected = 0;
assertEqual(actual, expected, desc);

desc = "with 00011110 should be 30";
input = "00011110";
actual = bin2dec(input);
expected = 30;
assertEqual(actual, expected, desc);

desc = "with 00101010 should be 42";
input = "00101010";
actual = bin2dec(input);
expected = 42;
assertEqual(actual, expected, desc);

desc = "with zero";
input = 0;
actual = dec2bin(input);
expected = "0";
assertEqual(actual, expected, desc);

desc = "with 30 should be 11110";
input = 30;
actual = dec2bin(input);
```

```
expected = "11110";
assertEqual(actual, expected, desc);

desc = "with 42 should be 101010";
input = 42;
actual = dec2bin(input);
expected = "101010";
assertEqual(actual, expected, desc);

function assertEqual(a, b, desc) {
  if (a === b) {
    console.log(`${desc} ... PASS`);
  } else {
    console.log(`${desc} ... FAIL: ${a} != ${b}`);
  }
}
```

Python

```
import unittest

def bin2dec(binStr):  # binStr is in str type
    bits = list(binStr)
    decVal = 0
    counter = 0

    for i in reversed(bits):
        decVal += 2 ** counter * int(i)
        counter += 1
    return decVal
    # returning value should be in int type.
    pass

def dec2bin(decNum):  # decNum is int type
    if decNum == 0:
        return str(0)
    binVal = ""
    while decNum:
        binVal += str(decNum % 2)
        decNum >>= 1
    return binVal[::-1]
    # returning value should be in str type.
    pass

# *** Tests ***

class Test(unittest.TestCase):
    def test_bin2dec_zero(self):
        desc = "with zero"
        inputNum = "0"
        actual = bin2dec(inputNum)
        expected = 0
        self.assertEqual(actual, expected, desc)

    def test_bin2dec_30(self):
        desc = "with 00011110 should be 30"
        inputNum = "00011110"
```

```python
        actual = bin2dec(inputNum)
        expected = 30
        self.assertEqual(actual, expected, desc)

    def test_bin2dec_42(self):
        desc = "with 00101010 should be 42"
        inputNum = "00101010"
        actual = bin2dec(inputNum)
        expected = 42
        self.assertEqual(actual, expected, desc)

    def test_dec2bin_zero(self):
        desc = "with zero"
        inputNum = 0
        actual = dec2bin(inputNum)
        expected = "0"
        self.assertEqual(actual, expected, desc)

    def test_dec2bin_30(self):
        desc = "with 30 should be 11110"
        inputNum = 30
        actual = dec2bin(inputNum)
        expected = "11110"
        self.assertEqual(actual, expected, desc)

    def test_dec2bin_42(self):
        desc = "with 42 should be 101010"
        inputNum = 42
        actual = dec2bin(inputNum)
        expected = "101010"
        self.assertEqual(actual, expected, desc)

unittest.main(verbosity=2)
```