# Problem Solving: Functions

Relevel
by Unacademy

# List of Problems Involved

- Function Calling and Reusability
- Sum of Prime
- Sum of Digits
- Power of Number
- Factor

# Function Calling and Reusability

**Problem Statement:**

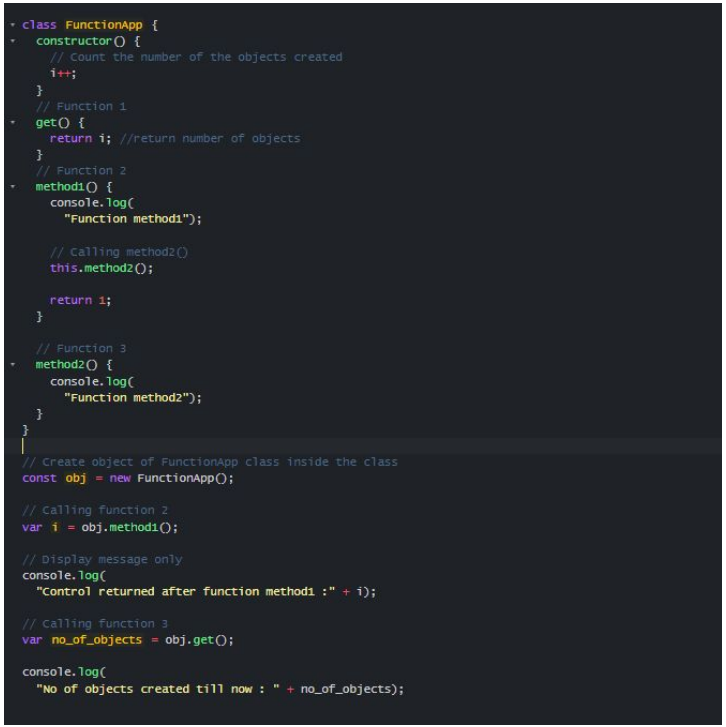
Create a class having multiple functions and showcase the function calling and reusability concept of the function

**Steps -**

1. Create a class FunctionApp
2. Create function 1 - to count number of objects created
3. Create function 2 - execute some logic
4. Create function 3 - execute some logic
5. Call function 2 from function 3
6. Functions can be called multiple times instead of writing logic again and again

# Function Calling and Reusability

**Code Link -** https://jsfiddle.net/86ptugav/



```
class FunctionApp {
  constructor() {
    // Count the number of the objects created
    i++;
  }
  // Function 1
  get() {
    return i; //return number of objects
  }
  // Function 2
  method1() {
    console.log(
      "Function method1");

    // Calling method2()
    this.method2();

    return 1;
  }

  // Function 3
  method2() {
    console.log(
      "Function method2");
  }
}

// Create object of FunctionApp class inside the class
const obj = new FunctionApp();

// Calling function 2
var i = obj.method1();

// Display message only
console.log(
  "Control returned after function method1 :" + i);

// Calling function 3
var no_of_objects = obj.get();

console.log(
  "No of objects created till now : " + no_of_objects);
```

# Sum of Prime

**Problem Statement:**

Given a number A which is an even integer, you need to print two prime numbers whose sum will be equal to A.

Constraint: A>2

Input - 4

Output - 2,2

**Explanation:**

For the range of values from 2 to A, start iterating the numbers from both the ends. Example if A is 8 then the range will be 2, 3, 4, 5, 6, 7

Since we need to find pair, Select the number `i` and `A-i`.

Check if both are prime or not. If both are prime then log the output and return.

We will iterate in the range till the middle range so as to keep position `i` <= `A-i` as comparison above. That will just be a repetitive task.

# Sum of Prime

Lets understand this by taking the above range as example:
Range = 2,3,4,5,6,7
A= 8
Let the pair be p1 and p2
**Iteration-1**
p1 = 2  & p2 = 6
2 is prime but 6 is not therefore continue
**Iteration-2**
p1 = 3  & p2 = 5
Both 3 and 5 are prime, hence log there output and return.

And to check prime, we are simply trying to find a factor of the number starting from 2 to square root of it. If there exists a factor, then the number is not prime.

# Sum of Prime

**Code Link -** https://jsfiddle.net/ckg4e5v6/

```
function isPrime(n) {
  for (i = 2; i <= Math.sqrt(n); i++) {
    if (n % i == 0) return false;
  }
  return true;
}

function sumPrime(n) {
  for ( j = 2; j <= n/2; j++) {
    if (isPrime(j) && isPrime(n - j)) {
      console.log(j + "," + (n - j));
    }
  }
}

var n = 15;
sumPrime(n);
```

# Sum of Digits

**Problem Statement:**
Given a positive number you need to write a program to find the sum of the inner digits of that number.
Input - 2124
Output - 3

**Explanation - 1:**
The Inner digits of the number 2124 are 1 and 2 whose sum is 3

**Explanation - 2:**
The Inner digits of the number 70 is None therefore the output will be -1.
Note: Same is the case for number with single digit

# Sum of Digits

**Solution Explanation:**

Since the internal digit can happen only on more than two digits, we first need to check that. If the condition fails, we simply return -1.

Now we need to perform the addition of the middle elements. To skip the rightmost element, we are first dividing it by 10.

So say if the number is 1234, then after dividing by ten, we are left with 123

i.e., on dividing by 10, we get rid of the leftmost digit of the number.

# Sum of Digits

Therefore in

**Iteration-1**

num =1234 / 10 = 123

sum  = 0 + 123 % 10 = 0+3 = 3

Since 123%100 = 3 we move to iteration 2

**Iteration-2**

num=123/10 = 12

Sum = 3 + 12%10 = 3+2 = 5

Now 12 %100 is 0 therefore we break the loop and return the sum. Which in this case is 5.

# Sum of Digits

**Code Link -** https://jsfiddle.net/fn9dab18/

```
function getCount(n) {
  var count = 0;

  while (n != 0) {
    count = count + 1;
    n = n / 10;
  }

  return count;
}

function getDigitSum(num) {


  var sum = 0;
  if (getCount(num) < 3) {
    return -1;
  }

  num = Math.floor(num / 10);
  sum = sum + num % 10;

  while (Math.floor(num / 100) != 0) {
    num = Math.floor(num / 10);
    sum = sum + num % 10;
  }
  return sum;
}



var n = 222314;
console.log(getDigitSum(n));
```

# Power of a Number

**Problem Statement:**

Given two numbers A and B,  You are supposed to find A to power B without using any built -in function.

Input - num=10, power=2

Output - 100

**Explanation:**

To compute the power of the number without using the in built function, we can simply multiply the number to itself the given number of times. I.e. 4 ^ 3 = 4 * 4 * 4 = 64.

# Power of a Number

**Code Link -** https://jsfiddle.net/x8sLmpn9/

```
function computePower( num,  power)
{
    var result = 1;

    while (power > 0) {
        result = result * num;
    power -=1;
    }
    return result;
}



  var number = 10;
  var power = 2;
  console.log(computePower(number,power));
```

# Power of a Number

**Optimized Solution:**

In the previous solution the while loop was iterating n number of time, where n was equal to the value of power. But in the optimized solution this iteration is reduced to half by  using a right shift operator for each iteration. Whenever the power value becomes odd, we can multiply the number with the resultant and store it in the resultant else; we simply update the number by multiplying it.

To check if number is even or odd We can perform a bit-wise operation. If the result is one, then the number is odd else even.

# Power of a Number

**Code Link -** https://jsfiddle.net/w7bqrx2m/

```
function computePower(num, power) {
  var result = 1;

  while (power > 0) {
    if ((power & 1) == 1)
      result = result * num;

    power = power >> 1;
    num = num * num;
  }
  return result;
}


var number = 10;
var power = 2;
console.log(computePower(number, power));
```

# Factor

**Problem Statement:**

Given a number A you are supposed to take the digits of that number, and find a permutation such that 60 becomes a factor of that permutation. If there exists such a permutation then return 1 else 0.

Input - 2340

Output - 1

# Factor

**Explanation:**

Given 2340, a possible permutation of its digit which is divisible by 60 can be 4320. This is equivalent to 60*72 therefore the result is 1.

60 is a composite number and any number which is divisible by composite numbers is supposed to be divisible by the prime number raised to the highest power.  Which in case of 60 are 4*5*3.

So any number that is divisible by 3,4,5 will be divisible by 60.

1. Now for  number to be divisible by 5, last digit is supposed to be 0 or 5
2. Number is divisible by 4 if its last two digits are divisible by 4.

On combining the above two points, we can conclude that any number which needs to be divisible by both 4,5 needs to have 0 as the last digit and that second last digit as even.

Finally to check if the number is divisible by 3, we need to sum the digits and see if it goes by 3. Using all these 3 we can come up with the solution to figure out if there exists a digit permutation which is divisible by 60.

# Factor

**Code Link -** https://jsfiddle.net/9kntswmo/

```
function isDivisibleBy60(num) {

  if (num % 10 == 0)
    return 1;

  var sum = 0;
  last = false;
  secondLast = false;
  while (num != 0) {
    i = num % 10;
    num = num / 10;
    if (i % 2 == 0 && (i > 0 || last))
      secondLast = true;
    if (i == 0)
      last = true;
    sum += i;
  }

  if (sum % 3 == 0 && last && secondLast)
    return 1;
  return 0;
}

var n = 2340;
console.log(isDivisibleBy60(n));
```

# Practice Questions

1. Given 2 numbers as input. You need to find if they have opposite signs or not.
   Input -> +123, -12
   Output -> true

2. Given 3 numbers. You need to find the smallest of three numbers without using comparison operators
   Input -> 12,44,2
   Output -> 2

# Thank You!