

Programming Fundamental



Exploring

#10 Algorithms

Fizz Buzz

- Fizz Buzz is an algorithm function that will log out to the console every number from 1 to “num”.
- For each number, if the number is divisible by 3, it'll log out the word ***“Fizz”*** instead of that number.
- Next, if the number is divisible by 5, it'll log out the word ***“Buzz”*** instead of that number.
- And finally, if a number is divisible by both 3 and 5, we want to logout the word ***“FizzBuzz”*** instead of that number.
- Beside is the result of ***fizzBuzz(20)***

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
```

Fizz Buzz

```
const fizzBuzz = (num) => {  
  for (let i = 1; i <= num; i++) {  
    if (i % 15 === 0) {  
      console.log('FizzBuzz');  
    } else if (i % 3 === 0) {  
      console.log('Fizz');  
    } else if (i % 5 === 0) {  
      console.log('Buzz');  
    } else {  
      console.log(i);  
    }  
  }  
};
```

```
fizzBuzz(20);
```

Fibonacci

Fibonacci sequence characterized by the fact that every number after the first two is the sum of the two preceding ones:

1, 1, 2, 3, 5, 8, 13, 21,
34, 55, 89, 144...

input= `fibonacci(6)` then output= 8

Fibonacci

```
const fibo = (urut) => {  
  if (urut < 3) {  
    return 1;  
  }  
  else {  
    return fibo(urut-1) + fibo(urut-2);  
  }  
}  
console.log(fibo(6));
```

Palindrome

Palindrome is a word or phrase that is spelled the same way both backward and forward. Note that we'll *ignore* any punctuation character such as commas, apostrophes, etc.

Example of Palindrome:

- Malam
- Katak
- Turut
- Asa
- Kakak
- Kasur rusak
- Race car
- Madam, I'm Adam

```
input=  
Palindrome('Asa')  
  
output=  
true
```

Palindrome

```
const Palindrome = (kata) => {  
  const karakter =  
    kata.toLowerCase().replace(/[^a-z]/g, '')  
    .split('');  
  
  if (karakter.join('') ===  
    karakter.reverse().join('')) {  
    return true;  
  } else {  
    return false;  
  }  
}  
  
const hasil = Palindrome("Ma1am");  
console.log(hasil);
```

Reverse Array In Place

This algorithm function will take in an array as a parameter, then it'll reverse that array and return us the reversed array.

`reverseArray([1,2,3,4,5,6,7,8])`

1, 2, 3, 4, 5, 6, 7, 8



8, 7, 6, 5, 4, 3, 2, 1

Reverse Array In Place

```
const reverseArray = (arr) => {  
  for (let i=0; i<Math.floor(arr.length/2); i++){  
    const tempArr = arr[i];  
    arr[i] = arr[arr.length - 1 - i];  
    arr[arr.length - 1 - i] = tempArr;  
  }  
  
  return arr;  
};  
  
console.log(reverseArray([1,2,3,4,5,6,7,8]));
```

Reverse Words

This algorithm function will take in a string as parameter, then it'll reverse every word in that string and return the new string.

`reverseWords('Hai aku Lintang')`

Hai aku Lintang



iaH uka gnatniL

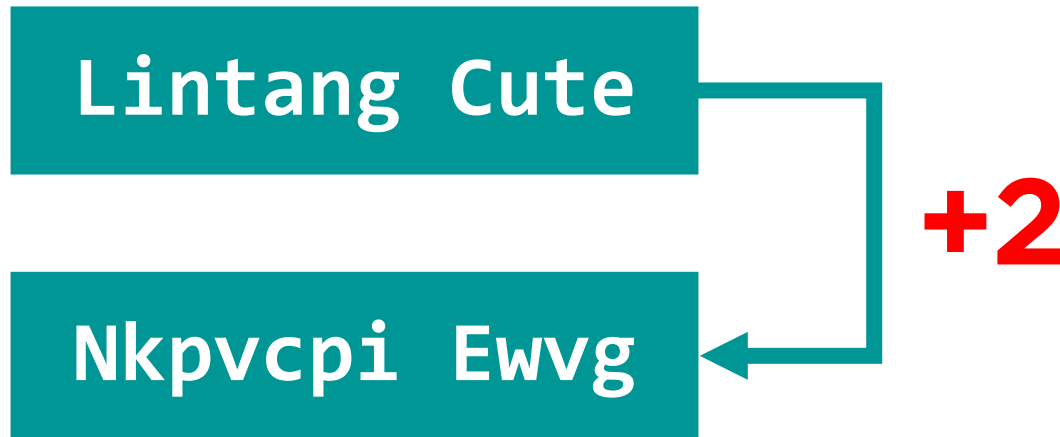
Reverse Words

```
const reverseWords = (string) => {  
  const wordsArr = string.split(' ');  
  let reversedWordsArr = [];  
  wordsArr.map((word) => {  
    let reversedWord = '';  
    for (let i = word.length - 1; i >= 0; i--) {  
      reversedWord += word[i];  
    }  
    reversedWordsArr.push(reversedWord);  
  });  
  return reversedWordsArr.join(' ');  
};  
console.log(reverseWords('Hai aku Lintang'));
```

Caesar Cipher

Caesar Cipher algorithm will take 2 parameters on its function: a string and a number. The objective of Caesar Cipher algorithm is to shift every letter in the given string by the number that is passed in.

`caesarCipher('Lintang Cute', 2)`



Caesar Cipher

```
const caesarCipher = (str, num) => {  
  num = num % 26;  
  const lowerCaseString = str.toLowerCase();  
  const alphabet = 'abcdefghijklmnopqrstuvwxyz'.split('');  
  let newString = '';
```

```
  for (let i = 0; i < lowerCaseString.length; i++) {  
    const currentLetter = lowerCaseString[i];  
    if (currentLetter === ' ') {  
      newString += currentLetter;  
      continue;  
    }
```

```
    const currentIndex = alphabet.indexOf(currentLetter);  
    let newIndex = currentIndex + num;  
    if (newIndex > 25) {  
      newIndex = newIndex - 26;  
    } else if (newIndex < 0) {  
      newIndex = newIndex + 26;  
    }  
    if (str[i] === str[i].toUpperCase()) {  
      newString += alphabet[newIndex].toUpperCase();  
    } else {  
      newString += alphabet[newIndex];  
    }  
  }  
  return newString;  
};
```

```
console.log(caesarCipher('A Zoo Keeper', 2));
```

Bubble Sort

```
var x = [6000, 34, 203, 3, 746, 200, 984, 198, 764, 9, 1]
const bubbleSort = (array) => {
  for (let i = array.length; i > 0; i--) {
    for (let j = 0; j < i; j++) {
      if (array[j] > array[j + 1]) {
        const temp = array[j];
        array[j] = array[j + 1];
        array[j + 1] = temp;
      }
    }
  }
  return array;
}
console.log(bubbleSort(x));
```

Mean, Median & Mode

- **Mean** is the average value of a dataset.
- **Median** is the middle number of a dataset.
- **Mode** is the most frequent number of a dataset.

[1,2,3,2,5,2,7,2]

Mean = 3 | Median = 2 | Mode = 2

Mean

```
var x = [ 1,2,3,2,5,2,7,2 ]
```

```
const getMean = (array) => {  
  let sum = 0;
```

```
  array.forEach(num => {  
    sum += num;  
  });
```

```
  const mean = sum / array.length;  
  return mean;  
}
```

```
console.log(getMean(x))
```


Median

```
var x = [ 1,2,3,2,5,2,7,2 ]
```

```
const getMedian = (array) => {  
  array.sort((a, b) => a - b);  
  let median;  
  if (array.length % 2 !== 0) {  
    median = array[Math.floor(array.length / 2)];  
  }  
  else {  
    const mid1 = array[(array.length / 2) - 1];  
    const mid2 = array[array.length / 2];  
    median = (mid1 + mid2) / 2;  
  }  
  return median;  
}
```

```
console.log(getMedian(x))
```

Mode

```
var x = [ 1,2,3,2,5,2,7,2 ]

const getMode = (array) => {
  var modeObj = {};
  // create modeObj
  array.forEach(num => {
    if (!modeObj[num]) {
      modeObj[num] = 0;
    }
    modeObj[num]++;
  });
  // create array of mode/s
  var maxFrequency = 0;
  var modes = [];
  for (let num in modeObj) {
    if (modeObj[num] > maxFrequency) {
      modes = [num];
      maxFrequency = modeObj[num];
    }
    else if (modeObj[num] === maxFrequency) {
      modes.push(num);
    }
  }
  // if every value appears same amount of times
  if (modes.length === Object.keys(modeObj).length) {
    modes = [];
  }
  return modes;
}

console.log(getMedian(x))
```