Time Complexity Analysis of JavaScript Functions

print()

```
function print() {
  console.log("Hello World");
}
```

Time Complexity: O(1)

sumArray()

```
function sumArray(arr) {
  let sum = 0;
  for (let i = 0; i < arr.length; i++) {
    sum += arr[i];
  }
  return sum;
}
```

Time Complexity: O(n)

findX()

```
function findX(arr) {
  let x = [];
  for (let i = 0; i < arr.length; i++) {
    for (let j = 0; j < arr.length; j++) {
      if (arr[i] + arr[j] === 10) {
        x.push([arr[i], arr[j]]);
      }
    }
  }
  return x;
}
```

Time Complexity: O(n^2)

getFirstTwoElements()

```
function getFirstTwoElements(arr) {
  if (arr.length < 2) {
    return null;
  }
  const first = arr[0];
  const second = arr[1];
  return [first, second];
}
```

Time Complexity: O(1)

processTwoArrays()

```
function processTwoArrays(arr1, arr2) {
  let sum1 = 0;
  for (const item of arr1) {
    sum1 += item;
  }
  let sum2 = 0;
  for (const item of arr2) {
    sum2 += item;
  }
  return sum1 + sum2;
}
```

Time Complexity: O(n + m)

countF()

```
function countF(n) {
  let count = 0;
  for (let i = 1; i < n; i = i * 2) {
    count++;
  }
  return count;
}
```

Time Complexity: O(log n)

Worst, average and best cases:

findElement()

```
function findElement(sortedArr, target) {
  for (let i = 0; i < sortedArr.length; i++) {
    if (sortedArr[i] === target) {
      return i;
    }
  }
  return -1;
}
```

Best: When target is at starting of array

Average: When target is at middle of array

Worst: When target is at end of array

recursiveSum()

```
function recursiveSum(n) {
  if (n <= 0) {
    return 0;
  }
  return n + recursiveSum(n - 1);
}
```

Best: When n is 0 or less than 0

Average: When n is slightly larger than 0

Worst: When n much larger than 0

dFunction()

```
function dFunction(arr) {
  const seen = {};
  for (let i = 0; i < arr.length; i++) {
    if (seen[arr[i]]) {
      return true;
    }
    seen[arr[i]] = true;
```

```
  }
  return false;
}
```

Time Complexity: Best: O(1), Worst: O(n)

repeatLog()

```
function repeatLog(arr) {
  for (let i = 0; i < arr.length; i++) {
    let repetitions = arr[i];
    for (let j = 0; j < repetitions; j++) {
      console.log('hello');
    }
  }
}
```

Time Complexity: O(2^n)