

Loop, Functions, and callbacks

Calculating the sum from 1 to 50

Loops

Dumb way

```
1 let ans = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 +  
  11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 19 + 20 +  
  21 + 22 + 23 + 24 + 25 + 26 + 27 + 28 + 29 + 30 +  
  31 + 32 + 33 + 34 + 35 + 36 + 37 + 38 + 39 + 40 +  
  41 + 42 + 43 + 44 + 45 + 46 + 47 + 48 + 49 + 50;  
2  
3 console.log(ans);  
4
```

Generate % I

Lot of code will be required to write from 500 or 5000

When we have to use repeated logic then we should use loop.

Loops

Better way - For loops

```
1  
2 let ans = 0;  
3  
4 for (let i = 1; i <= 50; i++) {  
5   ans = ans + i;  
6 }  
7  
8 console.log(ans);
```

In for loop, the whole line doesn't immediately run,
Initialisation take place first and only once(let i =1)

The comparison section runs ($i \leq 50$) if this condition is true we will loop in when the condition is not met we will break out of for loop (to line 8)

Then control reaches line 5 ($\text{ans} = \text{ans} + i$) , after line 5 (body of for loop) is executed it goes to $i++$ (update section) then it checks the condition, and then either loop body or terminate .

To visualize your code

<http://latentflip.com/loupe>

Functions

A function in JavaScript is a set of statements that performs a task or calculates a value it should take some input and return an output where there is some obvious relationship between the input and the output.

Example:

```
function sum (a,b) {  
    return a+b;  
}
```

```
console.log(sum(5,6));
```

Why do we need functions?

Functions

Why do we need functions?

```
1  
2 function findSum(n) {  
3   let ans = 0;  
4   for (let i = 1; i < n; i++) {  
5     ans = ans + i  
6   }  
7   return ans;  
8 }  
9  
10 let ans = findSum(100)  
11 console.log(ans);  
12  
13 let ans2 = findSum(1000)  
14 console.log(ans2);  
15
```

```
1  
2 let n = 100;  
3 let ans = 0;  
4  
5 for (let i = 1; i < n; i++) {  
6   ans = ans + i  
7 }  
8 console.log(ans);  
9  
10 let n2 = 1000;  
11 let ans2 = 0;  
12  
13 for (let i = 1; i < n2; i++) {  
14   ans2 = ans2 + i  
15 }  
16 console.log(ans2);  
17
```

Here without function, we can also do the same operations.

We need functions to prevent us from repeating the same logic

We violate **DRY** principle when we don't use functions.

Callback Functions

Basically calling function as argument

- Can we call one function inside another function ?

Yes!

Callback Functions

Step 1 - Can you call one function inside another function?
Yes

```
index.js > ...
1 // finds the square of the input
2 function square(n) {
3   return n * n
4 }
5
6 // finds the sum of the squares of the inputs
7 function sumOfSquares(a, b) {
8   const val1 = square(a);
9   const val2 = square(b);
10
11   return val1 + val2;
12 }
13
14 console.log(sumOfSquares(1, 2));
```

Now we want sum of cubes and sum of squares

```
function sq(n) {
  return n*n;
}
function cube(n) {
  return n*n*n;
}
function sumOfSq(a,b) {
  return sq(a)+sq(b);
}
function sumOfCube(a,b) {
  return cube(a)+cube(b);
}
```

```
}  
const ans = sumOfSq(2,3);  
console.log(ans);  
console.log(sumOfCube(2,3));
```

Is **DRY** being violated?

Callback Functions

```
index.js > f cube  
1  
2 function square(n) {  
3   return n * n  
4 }  
5 function cube(n) {  
6   return n * n * n  
7 }  
8  
9 function sumOfSquares(a, b) {  
10  const val1 = square(a);  
11  const val2 = square(b);  
12  
13  return val1 + val2;  
14 }  
15 function sumOfCubes(a, b) {  
16  const val1 = cube(a);  
17  const val2 = cube(b);  
18  
19  return val1 + val2;  
20 }  
21 console.log(sumOfCube(1, 2));
```

Is DRY being violated here?

Yes, Here comes the callback in the picture!

Callback Functions

```
index.js > f square  
1 function square(a) {  
2   return a * a  
3 }  
4  
5 function sumOfSomething(a, b, fn) {  
6   const val1 = fn(a);  
7   const val2 = fn(b);  
8   return val1 + val2;  
9 }  
10  
11 sumOfSomething(a, b, square)
```

Solution

```
function sq(n) {  
    return n*n;  
}  
function cube(n) {  
    return n*n*n;  
}  
function sumOfSomething(a,b,callback) {  
    console.log(callback);  
    return callback(a)+callback(b);  
}  
const ans = sumOfSomething(2,3,sq);  
console.log(ans);  
console.log(sumOfSomething(2,3,cube));
```

[Function: sq]

13

[Function: cube]

35

Anonymous function:

```
function sumOfSomething(a,b,callback) {  
    console.log(callback);  
    return callback(a)+callback(b);  
}  
  
const ans = sumOfSomething(2,3,function(n) {  
    return n*n;  
});  
console.log(ans);
```

[Function (anonymous)]

13