Loops

→ way to repeat task without writing code again and again.

→ we can run same code by writing again and again but it will increase the file size making it slower.

Syntax:

```
for(start; condition; change){
    ...
}
```

* run for loop 1-22

```
for(var i=1; i<23; i++){
    console.log(i);
}
```

* print 200-200 (reverse loop)

```
for(var i=200; i>99; i--)
{
    console.log(i);
}
```

* printing value outside loop

```
var i=1;

for(; i<=10; i++){
    console.log(i);          // 1-10
}
console.log(i+"fail");       // 11 fail
```

* loop without increment/decrement

```
for( var i = 1; i<=10;) {
        console.log(i);
   }
```

→ It will keep running as the condition is true forever.

we can also write:

```
for( var i = 1; ; ) {
        console.log(i);
    }
```

→ points i forever

→

```
for(;;) {
        console.log("Hlo");
    }
```

* i outside for loop

```
for( var i = 1; i<=10; i++) {
        console.log(i);                    // 1-10
   }
console.log("outside loop:", i);           // outside loop 11
```

→ We can console/access the i, outside the loop because it is functional scoped.

→ let won't be accessible outside loop

Sum of n natural numbers

```
const prompt = require("prompt-sync")();
let number = Number(prompt("Enter number "), 10);

if (isNaN(number)) {
        console.log("Provide number dude");
    }
    else {
        if (number < 0) {
            console.log("Provide Positive number dude");
        } else {
            let sum = 0;
            for (let i = 1; i <= number; i++) {
                sum += i;
            }
            console.log("sum is", sum);
        }
    }
```

factorial of number

main logic

```
else {
    let factorial = 1;
    for (let i = 1; i <= number; i++) {
        factorial *= i;
    }
    console.log("Factorial is", factorial);
}
```

→ 5 → 5x4x3x2x1 = 120

initial should not be 0 → instead 1

finding factors

Basic way:

```
if (num > 0) {
    for (let i = 1; i <= num; i++) {
        if (num % i === 0) {
            console.log(i);
        }
    }
}
```

→ How can we make our code efficient, we have to reduce number of iterations.

see, the factors of 36 → 1, 2, 3, 4, 6, 9, 12, 18, 36

except 36 itself all numbers are less than half of 36 or equal to

So we can write as,

```
if (num > 0) {
    for (var i = 1; i <= Math.floor(num/2); i++) {
        if (num % i === 0) {
            console.log(i);   // upto half
        }
    }
    console.log(num);   //number itself
}
```

* Another optimization with square root

Factors of 36 →
→ 1 x 36
→ 2 x 18
→ 3 x 12
→ 4 x 9
→ 6 x 6

Factors of 18
→ 1 x 18
→ 2 x 9
→ 3 x 6
→ 4

Square root of 36 → $\sqrt{36}$ → 6

Square root of 18 → $\sqrt{18}$ → 4.xx

In both conditions, we can run the loop upto square root of number and pass get the factors in pairs.

```
if (isNan(number) || (number <0)){
        console.log ("Provide valid number");
    }
else {
    console.log ("The possible factors are:");
    let sqaRoot = Math.floor (Math.sqrt(number));

for( let i=1; i<=sqaRoot; i++) {          → loop runs minimal
    if(number% i === 0) {
        console.log(i);                   → 1st pair fact.
        if (i !== number% i) {
            console.log(number /a i);     → 2nd pair factor
                                              (if applicable)
    }
}
}
```

↓
This condition verifies no perfect square is written twice

sqaRoot of 25 → 5 at i=5 → 5 === 5

at i=1 → 1, 25 ... 5, 25 ↓ same program ↓ removes further

Find prime number between 1 & given number

```
let number = Number(prompt("Enter number"));

if (isNan(number) || (number<0)){
        console.log("Provide natural number");
}
else{
    console.log("The prime number upto given number
                are:");
    for (let i=2; i<=number; i++){
                    → starting from 2 as 1 is not prime
        let prime = true;
        for (let j=2; j<=Math.floor(Math.sqrt
                    i++){
            if(i%j===0){
                prime = false;
                break;
            }
        }
        if (prime){
            console.log(i);
        }
    }
}
```

* function to check prime or not. (optimised)

```
function isPrime(n){
        if (n<1) return false; // 1 is not prime
        if (n===2) return true; // 2 is prime
        if (n%2===0) return false; // even number are
                                   prime
```

incrementing by only odd ↑

```
for (let i=3; i<=Math.floor(Math.sqrt(n)); i+=2) {
        if(n%i===0) return false
    }
    return true;
}
```

break and continue

Inside loop if we use break → loop won't iterate again.

If we use continue, loop won't ~~continue~~ the code after continue will not execute for the given iteration, but loop will continue to iterate from next iterable.

while loop

```
let i=2;
while (i<10)
  {
    console.log(i);
    i++
  }
```

→ When we know iterations, that how many times we have to run → use for loop

→ When we have to keep running loop until certain situation meets, we use while loop.

Sum of digits

555 → 15

555 % 10 → 5 >> Math.floor(555/10) → 55 >> 55 % 10 → 5 >> Math.floor(55/10) >> 5 % 10 → 5 >> 5/10 → 0

main logic:

```
let sum = 0;
while (number > 0) {
    sum += number % 10;
    number = Math.floor(number / 10);
}
console.log(sum);
```

5	10	15
55	5	0

Reverse a number

1234 → 4321

To do so, we need to extract the last digit of the number and our loop reverse = reverse * 10 + last digit of number, until the number is 0.

```
let reverse = 0;
while (number !== 0) {
    reverse = reverse * 10 + number % 10;
    number = Math.floor(number / 10);
}
console.log("reversed num", reverse);
```

check the Strong Number

→ sum of the factorials of the digit is equal to the original number.

145 → 1! + 4! + 5!
= 145 (strong number)

```
const prompt = require("prompt-sync")();
let number = Number(prompt("Enter the number : "));

function factorial(num) {
    if (num === 1 || num === 0) return 1;
    return num * factorial(num - 1);
}

if (isNaN(number) || (number < 0)) {
    console.log("Provide a natural number");
} else {
    let originalNumber = number;
    // storing original number for comparision
    let sum = 0;

    while (number > 0) {          // get last number and add its factorial
        sum += factorial(number % 10);
        number = Math.floor(number / 10);
    }
    if (sum === originalNumber) {
        console.log("It is strong number");
    } else {
        console.log("It is not strong number");
    }
}
```

do while Loop

```
do {
    ....        → at least runs once
}
while (condition);
```

Guess the Number

```
const prompt = require("prompt-sync")();
const randomNum = Math.floor(Math.random() * 100 +1

let guessed;

do {
    guessed = Number(prompt("Guess the number b
                1 and 100: "), 10);
    if (guessed >=100 || guessed <1 || isNan(guessed
        console.log("Try between 1 and 100");
    } else {
        if (guessed < randomNum) {
            console.log("Too low");
        } else if (guessed > randomNum) {
            console.log("Too high");
        } else {
            console.log("Correct")
        }
    }
} while (guessed !== randomNum)
```