# Repetition

Liberation is when you break out of the loop.

# Lesson Objectives

- Understand repetition control logic
- Learn to use JavaScript loop syntaxes

# Control logic

- Sequence
- Selection
- **Repetition**

# Repetition/ Looping

- During code execution, some statements need to be executed more than one time. For achieving this, programming languages provide loop syntax, which keeps on executing the statements inside the loop until certain condition is met.
- Loop can be controlled based on a counter or based on some event (sentinel controlled).
- Based on when the looping condition is checked, loop can be pre-test or post-test loop.

# while loop

- pre-test loop
- Can be used as both counter-controlled loop or sentinel-controlled loop.

```
while (condition) {
        statements;
}
```

- Counter-controlled
  - see example: *lecture_codes/lesson3/while_counter_controlled.js*
- Sentinel-controlled
  - see example: lecture_codes/lesson3/while_sentinel_controlled.js

# do while

- post-test loop
- Better suited for sentinel-controlled loop
  - The statements inside do block is guaranteed to be executed at least once
- Example – updated sentinel controlled while loop.
  - See lecture_codes/lesson4/do_while.js
  - Notice: this version makes prompt at only one place, that is inside the do part of the do-while loop.

# for

- Pre-test loop
- Better suited for counter-controlled loop
  - Updating counter is part of the syntax, hence you won't forget!
- Careful! if you forget to update the loop counter or because of logic error loop counter is never reaching the exit condition, loop will be infinite.

```
for (initialization; condition; update) {
        statements;
}
```

- Example - updated counter-controlled while loop example
  - See *lecture_codes/lesson4/for_loop.js*

# Exercise

- Write a loop that prints all even number between 1 to 20 (inclusive)
  - Write both `while` and `for` versions
- Write a loop that keeps on prompting for age until you enter age older than 18
  - Write both `while` and `do while` versions.
- Make "Compound Interest" example from chapter 8, working on your machine.

# break and continue

- The break statement ends a loop prematurely.
  - Example - finding if a number is prime without using break statement.
    - See *lecture_codes/lesson4/test_prime_no_break.js*
  - Example - finding if a number is prime using break statement.
    - See *lecture_codes/lesson4/test_prime_using_break.js*
- The continue statement "jumps over" one iteration in the loop.
  - See example: *lecture_codes/lesson4/continue_keyword.js*

# Main Point

- Looping/ repetition is one of the key control structure that adds a lot of power to computer programming. In most cases a loop should be finite. Programmer should be careful to avoid infinite loops. *Science of consciousness, When a programmer is thinking clearly, these kinds of mistakes will be avoided automatically. With a regular practice of TM, our body and mind becomes stress free and we will be able to think more clearly.*

# Nested Loops

- Loop inside of a loop

```
for(let i = 1; i<=10; i++){
    let row="";
    for(let j=1; j<=10; j++){
        row += i*j +"\t";
    }
    console.log(row);
}
```

# Exercise

- Write code to print following patterns on the console:

```
11111
22222
33333
44444
55555
```

```
12345
12345
12345
12345
12345
```

# Assignments

- Reading chapter 8
- Programming Assignments from chapter 8 (except question 11)
- Write program to print following patterns

```
1
22
333
4444
55555
```

```
1
12
123
1234
12345
```

```
55555
4444
333
22
1
```