10 May 2025 → Saturday
11 am

Const day = " ";

"string" → " 10 May 2025"

1. which day we are in
2. Can also calculate week we are in in the year 2025

Current = 139th

day = 138 / 7 → Week 19 (Math.floor) (Round closest int)

= 138 % 7 = 5 (Mon / Tues. Wed / Thur → Frid).

if (day === 1) { return "Monday" }
Tuesday.
. . .

7 % 7 = 0 to 6 → Monday to Sunday.

if (day =

num = 0

// Return Week day - Mon Tues. . .

if switch (num) {
case 0 : return break ;
case 1 :
default :- Not good input num

365 day → 139th

→ 139 % 7 → 5 → Saturday.

10 May 2025 →
↳ Jan Feb Mar Apr

Iterativalisive statements

for loops | while | do {

for (var initialization; condition; increment/decre) { stmt ; }

initialization
while (condition) { stmt ; Incr/decr }

initialization
do { stmt ; Incr/decr } while (condition)

Pattern:

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

for → 1 to 5
while →
do while →

for row 5 to 0 dec.
  for col 0 to row.
    print("*");

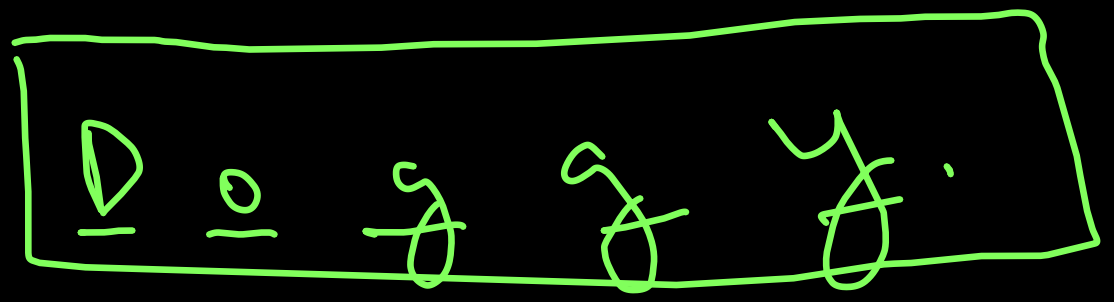
5 → 0 1 2 3 4 5
    0 1 2 3 4

→ while & for

do { stmt ; } while(condition);

D o g g y. → Password:

Input from the user



about :

original_pwd = "Cloggy"

enter_pwd = prompt('Enter pass')

if

Data Types

1. Array → Continuous collection of Data

push . pop . length , to String

const arr = [1 , 'a' , 'Praneeth' ; 10.5]
index : 0 1 2 3

→ Accessing elements o[n]

2. Hash Table :

Const details = {
name : 'Praneeth' , lastname : P
age : 30 ;
celebrate : ' Test '
fullName : func() {
this . name + this.last nam
}.

}

→ Key helps accessing data faster
→ Hash Table → inside another hash Table

Array Method :

- find I ndex . filter. . join . push & pop. .	. reduce. . slice . splice.

try in 10 min

. find Index → Indx

filter → functn

push & pop

join

zone.

7 , 8 , 9.

(1 , 3 , 5)

filter → functn check value > length.

values + / —

slice — arr (start, end) ⇒ shallow

splice — over (index, deletecount, delete insert, insert 2, ... insert n)

⇒ not mutate original array.

→. Shallow & Deep Copy.

. Reduce Method →.

(accumulation, current Value) => { =

}, startofaccumulator value)

→. [1, 2, 3, 4, 5]. reduce

((acc, cur) => { }, 0)

return acc.. → acc = acc + cur.