Programming 101

⇉ Declaring a variable → Const → Can't be changed later
→ let → Can be changed

⇉ Storing in a Variable →
```
let name = "Akshay";   → String Data type
let age = 23;          → Number data type
let isTeacher = true;  → Boolean
```

⇉ Printing something on Console →
```
Console.log(age);   → O/P: 23
```

⇉ String Concatenation →
```
Console.log(name + age)   → Akshay23
```
(Internally JS convert Num into str in this case)

⇉ Code execution happens line by line.

⇉ To store multiple values we can use Array
```
let Arr = [ 0, 10, 20, 40, 14];
```
→ Can create string array, any datatype (mix of datatypes)

⇉ To access array

index: 0 1 2 3 4

```
Arr[1]   → O/P: 10
```

⇉ If we access element which is not present in Array → `Arr[5]` O/P: undefined

⇉ We can store array inside an array
```
arr = [ 10, 20 [30, 40]]
arr[2] [1]   → O/P: 40
```

index	Value
0	10
1	20
2	[30, 40]

⇉ Objects in JS:

```
let obj = {
    firstName: "Akshay",
    lastName: "Saini",
    age: 23,
    isTeacher: true
}
```
Key → firstName; Value → "Akshay"

Contains Key : Value pairs, can be Accessed with . notation.

Key	Value
firstName	"Akshay"
lastName	"Saini"
age	23
isTeacher	true

```
console.log(obj.firstName
+ obj.lastName);
```
O/P: AkshaySaini

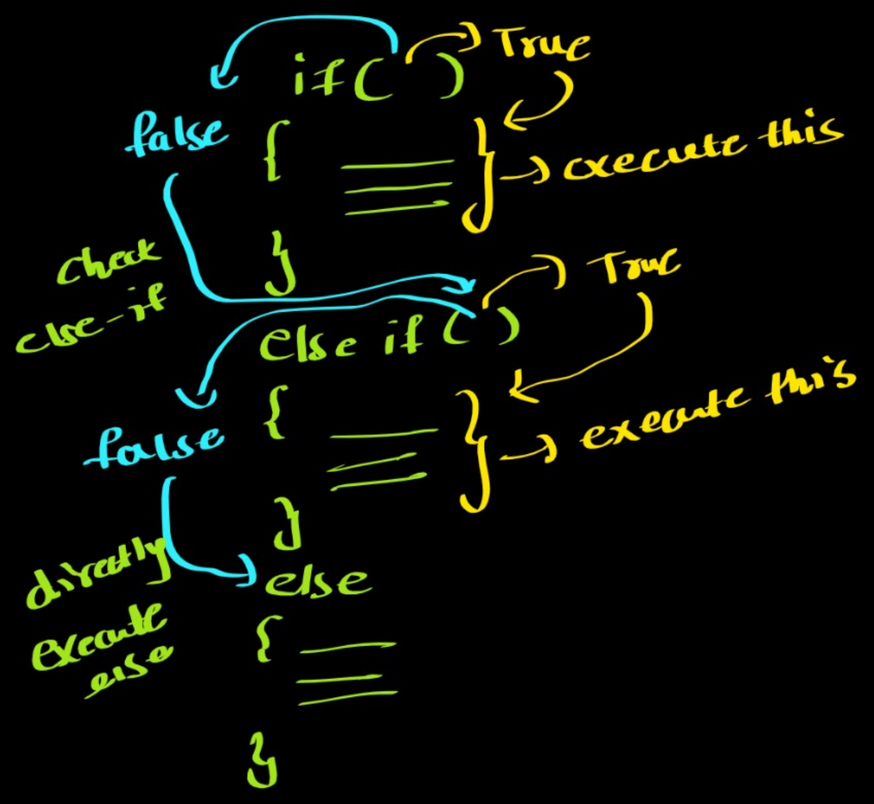
Function, if-else

⇉ function definition:
```
function functionname()
{
    ===   → what we want to do
    return _;   → return somevalue
}
```

⇉ If we didn't use return keyword (or) didn't mention any value after return it will return undefined.

⇒ To execute any function, we need to call the function → functionname()
(imp to call)

⇒ whenever we want to check some condition to execute at code then we use conditionals (if, else-if, else)



⇒ If we have only two conditions we can use (if-else)

⇒ for multiple conditions we can use (if - else-if - else).

Loops

① initialization, condition, change

```
for(let i=0; i<=4; i++)
{
    Console.log("Hello world");
}
```

Console:

i=0 ; Hello world
i=1 ; Hello world
i=2 ; Hello world
i=3 ; Hello world
i=4 ; Hello world
} 5 times

i=5 X false

②

```
for(let i=3; i<5; i++)
{
    console.log("Hw");
}
```

Console:

i=3 ; Hw
i=4 ; Hw
} 2 times

i=5 X false

③

```
for(let i=0; i<9; i=i+2)
{
    Console.log("Hw");
}
```

Console:

i=0 ; Hw
i=2 ; Hw
i=4 ; Hw
i=6 ; Hw
i=8 ; Hw
} 5 times

i=10 X false

④

```
for(let i=5; i>0; i--)
{
    console.log("Hw");
}
```

O/P:

i=5 ; Hw
i=4 ; Hw
i=3 ; Hw
i=2 ; Hw
i=1 ; Hw
} 5 times

i=0 X false

⑤
```
for(i=5 ; i<4 ; i++)
{
    console.log("Hw");
}
```
Console:
i=5 X false } 0 times

⑥
```
for(i=1 ; i>0 ; i++)
{
    console.log("Hw");
}
```
→ Infinite loop

* **while loop:**

```
let i=0;          → Initialization
while(i<5)        → condition
{
    console.log("Hw");
    i++;          → change
}
```

Console:
i=0 ; Hw
i=1 ; Hw
i=2 ; Hw
i=3 ; Hw
i=4 ; Hw
} 5 times
i=5 X false

Loops-2

```
function largestElement(arr)
{
    let largest = -Infinity;   // the smallest possible value
    for(let i=0 ; i<arr.length ; i++){
        if(arr[i] > largest)
            largest = arr[i]
    }
    return largest;
}
```

```
function smallestElement(arr)
{
    let smallest = Infinity;   // the largest possible value.
    for(let i=0 ; i<arr.length ; i++)
    {
        if(arr[i] < smallest){
            smallest = arr[i];
        }
    }
    return smallest;
}
```

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

```
function searchElement(arr, x)
{
    for(let i=0; i<arr.length ; i++)
    {
        if(arr[i]==x)
            return i;
    }
    return -1
}
```

```
SecondLargest(arr)
{  if(arr.length<2)
       return null
```
→ Covers edge case of arr being empty

```
   firstlargest = -infinity;
   secondlargest = -infinity;
   for(i=0; i<arr.length ; i++)
   {
      if(arr[i] > firstlargest)
      {
          Secondlargest = firstlargest;
          firstlargest = arr[i];
      }
      else if(arr[i] > secondlargest && arr[i] != firstlargest)
      {
          Secondlargest = arr[i];
      }
   }
   return secondlargest;
}
```
→ Covers duplicate elements edge case.

Loop in Loop

①
```
for(i=0; i<3 ; i++)
{
   for(j=0; j<3 ; j++)
   {
      Console.log( i, j);
   }
}
```

i	j
0	0
	1
	2
1	0
	1
	2
2	0
	1
	2

o/p:

0,0
0,1
0,2
1,0
1,1
1,2
2,0
2,1
2,2

→ Total 9 times

②
```
for(i=0; i<3 ; i++)
{
   for(j=0; j<i ; j++)
   {
      log(i,j)
   }
}
```

i	j
0	X
1	0 → 1 time
2	0 } → 2 times
	1 }

→ 0 times

o/p:

1,0
2,0
2,1

③
```
for(i=0 ; i<3 ; i++)
{
  for(j=0 ; j<=i ; j++)
  {
    log(i,j)
  }
}
```

i	j
0	0 → 1time
1	0 → 2timg
	1
2	0 → 3times
	1
	2

Console:

0,0
1,0
1,1
2,0
2,1
2,2

} 6 times

④
```
for(i=0 ; i<3 ; i++)
{
  for(j=i ; j>0 ; j--)
    log(i,j);
}
```

i	j
0	X
1	1
2	2

O/P:

1,1
2,2
2,1

④
```
for(i=0 ; i<3 ; i++)
{
  for(j=i ; j>=0 ; j--)
    log(i,j);
}
```

i	j
0	0
1	1
	0
2	2
	1
	0

Console:

0,0
1,1
1,0
2,2
2,1
2,0

```
for(i=5 ; i>0 ; i--)
{
  for(j=0 ; j<i ; j++)
    log(i,j)
}
```

i	j
5	0
	1
	2
	3
	4
4	0
	1
	2
	3
3	0
	1
	2
2	0
	1
1	0

Console:

5,0
5,1
5,2
5,3
5,4
4,0
4,1
4,2
4,3
3,0
3,1
3,2
2,0
2,1
1,0

} 15 times

Star patterns

n=4

☆ ☆ ☆ ☆
☆ ☆ ☆ ☆
☆ ☆ ☆ ☆
☆ ☆ ☆ ☆

```
for(i=0 ; i<n ; i++)
{   let row="";
    for(j=0 ; j<n ; j++){
        row = row + "☆";
    }
    Console.log(row);
}
```

row	i	j
☆☆☆☆	0	(0 to 3) 4 times
☆☆☆☆	1	(0 to 3) 4 times
☆☆☆☆	2	(0 to 3) 4 times
☆☆☆☆	3	(0 to 3) 4 times

n=4

*
* *
* * *
* * * *

```
for(i=0; i<n ; i++)
{
   let row="";
   for(j=0; j<i+1 ; j++)
   {
       row = row + "*";
   }
   log(row);
}
```

(or) j <= i

row	i	j
*	0	1 times (0<1) (or) (0<=0)
* *	1	2 times (0<2) (0<=1)
* * *	2	3 times (0<3) (0<=2)
* * * *	3	4 times (0<4) (0<=3)

n=5

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

```
for(i=0 ; i<n ; i++)
{
  let row = "";
  for (j=0 ; j<=i ; j++)
  {
       row = row + (j+1);
  }
  Console.log(row);
}
```

row	i	j
1	0	1time
1 2	1	2times
1 2 3	2	3 times
1 2 3 4	3	4 times
1 2 3 4 5	4	5 times

n=5

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

```
for(i=0 ; i<n ; i++)
{
  let row = "";
  for (j=0 ; j<=i ; j++)
  {
       row = row + (i+1);
  }
  Console.log(row);
}
```

row	i	j
1	0	1time
2 2	1	2times
3 3 3	2	3 times
4 4 4 4	3	4 times
5 5 5 5 5	4	5 times

n=5

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

```
for(i=0 ; i<n ; i++)
{  row="";
   for( j=0 ; j<n-i ; j++)
   {
       row = row + (j+1);
   }
}
```

row	i	j
1 2 3 4 5	0	5 times
1 2 3 4	1	4 times
1 2 3	2	3 times
1 2	3	2 times
1	4	1 time

n=5

* * * * *
* * * *
* * *
* *
*

```
for(i=0 ; i<n ; i++)
{  row="";
   for( j=0 ; j<n-i ; j++)
   {
       row = row + "*";
   }
}
```

row	i	j
* * * * *	0	5 times
* * * *	1	4 times
* * *	2	3 times
* *	3	2 times
*	4	1 time

n=5

```
    *
   * *
  * * *
 * * * *
* * * * *
```

```
for(i=0 ; i<n ; i++)
{   row = "";
  for(j=0 ; j<n ; j++)
  {
     if (j < n-(i+1))
          row = row + " ";
     else
          row = row + "*";
  }
  console.log(row);
}
```

row	i	j
`    *`	0	5 times
`   * *`	1	5 times
`  * * *`	2	5 times
` * * * *`	3	5 times
`* * * * *`	4	5 times

n=6

```
1
10
101
1010
10101
101010
```

```
for(i=0 ; i<n ; i++)
{   row = "";
  for(j=0 ; j<=i ; j++)
  {
     if (j%2 ==0)
          row = row + "1";
     else
          row = row + "0";
  }
}
```

row	i	j
1	0	1 time
10	1	2 times
101	2	3 times
1010	3	4 times
10101	4	5 times
101010	5	6 times

n=6

```
1
01
010
1010
10101
010101
```

```
toggle=1
for(i=0 ; i<n ; i++)
{   row = "";
  for(j=0 ; j<=i ; j++)
  {
     row = row + toggle
     if(toggle==1)
          toggle=0
     else
          toggle=1;
  }
} console.log(row);
```

row	i	j
1	0	1 time
01	1	2 times
010	2	3 times
1010	3	4 times
10101	4	5 times
010101	5	6 times

Count Digits

num = 1268 → O/P: 4 digits

num	count
1268	0
126	1
12	2
1	3
0	4

```
let count = 0
while (num>0) {
     num = num/10;
     count++
}
return count;
```

General Algo

Math.abs() → absolute value
removes negative

Math.floor() → one step down (removes decimal part)
Math.ceil() → one step ahead (next number)
Math.round() → ≥0.5 (next number)
<0.5 (same number)

Palindrome

⇒ last digit ⟶ num % 10

remove last digit ⟶ num = num/10;

```
function palindrome(x)
{   if(x<0) return false;
    xCopy = x;
    rev = 0;
    while(x>0)
    {
        rev = (rev x10) + (x%10)
        x = Math.floor(x/10);
    }
    return rev === xCopy;
}
```

Algo:
1. Save a copy of original number
2. reverse an integer and store it
3. Compare the reversed number with saved copy.
4. If same return true, else false.
5. for -ve numbers return false.

Reverse Integer

```
function reverse(x)
{
    let xcopy = x;
    x = Math.abs(x);
    let rev = 0;
    while(x>0)
    {
        rev = (rev x10) + (x%10)
        x = Math.floor(x/10);
    }
    let limit = Math.pow(2,31);
    if(rev < -limit || rev > limit-1) return 0;
    return (xcopy<0) ? -rev : rev;
}
```

1. Save copy of original
2. convert -ve to +ve if x is -ve.
3. reverse the numbers
4. if original is -ve return -ve of reverse else just return reverse.