Day 5

Function In JS

Block of code that performs a specific task, can be invoked(call) whenever needed.

=> Declared function

syntax :-

```
function functionName()
{
    // ....
}

function function name (param1, param2)
{

}

// function call

functionName();
```

redundancy -> repeat :-

```
function sum (x, y)
{   console.log(x+y);
}
```

return the value u can return array, string any number u want

```
function sum (x , y)
{
    s = x + y;
    return s;
}

let val = sum (3, 4)
    console.log (val);
```

In the function the parameter act like a local variable mean its is only in the scope

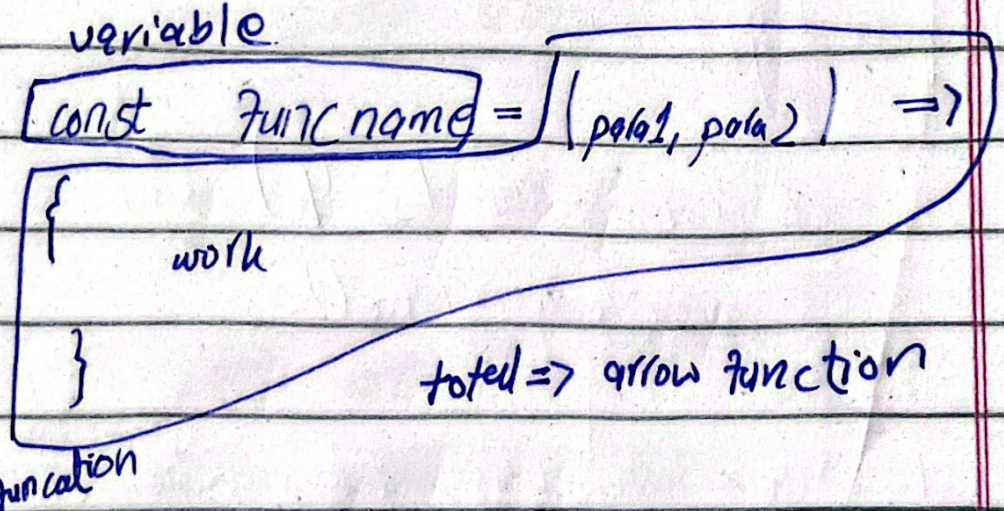
```
function sum (x, y)
{   //          ⟹ scope
}
```

You cannot access the parameter of parameter out of his scope.

fnn para → like local param
→ block scope of function.

⇒ Arrow Functions8- //modern Is

compact way of writing a function.



=>

```
const arrowSum = (a, b) => {
    console.log (a+b);
}
```

function variable

```
const mul = (a, b) =>
{
    mul = a*b
    return a*b
```

```
const print = ( ) => {
    console.log ("hello");
};
```

or `const print = ( ) => console.log("hello");`

let's Practise

```
let str = a, e, i, o, u

function vowels ( str )
{
    vowels
    if ( str == 'a, e, i, o, u')
        continue
        count++;
}

vowels ("hello");
```

count of vowels.

```
function count (str)
{   let count=0;
  for ( const char of str)
  {
    if ( char ==='a'||"e"||'i'||'o'||'u
    ) {
      count++;
      }
  }
  console.log(count);
  } }   return count;
```

By Arrow =>

```
const low = ( str ) => {
{
  //    //
}
```

=) ForEach loop in Arrays

aik aik index pay ja kay aik aik value

Methods — Function kay liyay

"abc".toUpperCase();

string — methods

usi operation ko perform kerwana.

ForEach is a function but this is connect with array so we can called method. method basically function but method is connect with object or Data structer bindy the method

=) Syntax

arr.forEach(callBackFunction)

Function in Js can pass as parameters & also we can return me function value

For Example:-

```
function abc ()
{
    console.log ("hello");
}
function myfun (abc)
{
    return abc;
}
```

In Js me function pass and return as as variable

call back function :- Here it is a function execute for each element in the array

A callback function is a function passed as an

argument to another function.

Example

```
let arr = [1,2,3,4,5]

arr.forEach(function printVal(val){
    console.log(val);
})
```

// each value of each index

Arrow function

```
arr.forEach((val) => {
    console.log(val);
    val.toUpperCase();
});
```

=> Call back have three parameters in arrow function

value, index, array
item position itself

```
arr.forEach(val, idx, arr)
    console.log(val.toUpperCase(), idx, arr)
});
```

Theorical concept
Interview Question

Higher order function/method

ForEach calls Higher order function/method.

function ko parameter kay zariay pass ker sky ya phir function return kerwa sky ya.

Practice. Question:-

Let arr = [2, 3, 5, 7]

~~arr.forEach(~~
function square (arr)
{ ~~for (let key in arr)~~
~~console.log (arr[key])~~
console.log (arr * arr)
}.

```
const square = (arr) =>
{
    console.log(arr**2)
}
arr.forEach(square);
```

Imp arrays Method

Map is very similar to forEach and the difference b/w map & for_each is that map retur new array.

Map => return New Array creates a new array with results of some operations the value its calledback return are used to form new array.

```
arr.map(callbackFunn(value, index, array)
{
    ;
}
```

```
let arr = [ 1, 2, 3 ];

arr.map( (val) =>
{
    console.log( val );
});
```

when we return the value by map in the return case the value return in the new arrays.

```
let newArr
    = arr.map( (val) =>
    {
        return val;
    }
    console.log ( newArr);
```

Array Methods:-

Filter:-

Creates a new Array of elements that gives true for a condition / filter

all even elents.

```
let newArr = all.filter ((val) =>
{
    return val % 2 == = = 0;
});
```

→ true store in newArray.

false

⇒ Reduce Method

Performs some operation & reduce the array to a single value. it returns that single value.

Input have = [2, 4, 6, 8]

return into single value.

like sum ⇒ avg

!!

24

6

20/4 = 5

E.g :-

```
const arrayL = [1, 2, 3, 4];

const

let output = ar.reduce ( previousvalue, currentvalue
                        => {

        return curr + pre;
});      console.log (output);
```

[1, 2, 3, 4]
pre ↑ curr ↑

res => 1 | 3 | 6
curr => 2 | 3 | 4

largest number in an Array

```
    return pre > curr ? per : curr;
});
```