JavaScript : Module ⑤ (Functions)

__/__/____

•> Functions in JS are more than just code containers - they are core tools for abstraction, reusability and scalability.

A Function is a block of code that performs a specific task. It allows you to encapsulation logic & reuse it throughout your program.

ex : Function greet (name) {
console.log (`Helo, ${name}!`); }
greet ("Alice"); // Helo, Alice !

•> Use

① Reusability - write once, use multiple times.
② Modularity - Break large problems into smaller.
③ Abstraction - hide complex logic behind simple name.
④ Testability - Functions make code easier to test & debug.

Types of Functions

① Function Declaration

- It is a simple way to define a named function using the function keyword. It's hoisted, meaning it can be called before it's defined.

```
function sum(a, b) {
  return a+b; };
```

__/__/__

② Function Expression

- It involves assigning a function to a variable. Unlike declarations, function expressions are not hoisted.

```
const divide = function (a, b) {
    return a / b;   };
```

③ Arrow Function (ES6)

- Are shorter syntax for writing functions, introduced in ES6. They do not have their own this and are ideal for concise operations.

```
const multiply = (a, b) => a * b;
```

④ Parameter vs Argument :

•) Parameter - A named variable in a function definition - acts as a placeholder.

•) Argument - the actual value passed to the function when it's called

```
Function greet (user) {      // user is a parameter
  console.log ("welcome, ${user}");
}

greet ("Emma");              → Argument
```

__/__/____

⑤ Call Back Function

- It is a function passed as an argument to another function, to be executed later - often used in asynchronous programming.

- useful for event handling, setTimeOut, and working with APIS.

```
function fetchData (callback) {
  setTimeOut (() => {
    callback ("Data loaded");
  }, 1000);
}
fetchData ((msg) => console.log (msg));
```

__/__/____

① Function 'makeTea' takes a type of tea as a parameter & returns a message like "Making green tea".

```
Function makeTea (typeOfTea) {
    return `Making ${typeOfTea}`;
}

let TeaOrder = makeTea ("green tea");
console.log (TeaOrder); // Making green tea
```

② Function 'orderTea' takes a tea type and has a nested function 'confirmOrder' that returns a confirmation message.

```
Function orderTea (teaType) {
    Function confirmOrder () {
    } return `Order Confirmed For ${teaType}`;

    return confirmOrder (); }
let orderConfirm = orderTea ("Chai");
console.log ( orderConfirm) ; // order confirmed for chai
```

③ Arrow function, 'calculate total' takes price and quantity, & returns the total cost. */

```
const calculateTotal = (price, quantity ) => price *
                                    quantity ;
let totalCost = calculateTotal (499, 100) ;
console.log ( totalCost ) ;        // O/P : 49900
```