**Different Types of functions**

1. Named Function

A function with a specific name that can be reused.

function greet(name) {

return "Hello, " + name + "!";

}

console.log(greet("Alice"));

===============================================

2. Anonymous Function

A function without a name, often assigned to a variable.

let sum = function(a, b) {

return a + b;

};

console.log(sum(5, 3));

==============================

3.Arrow Function

A shorter syntax for writing functions.

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

console.log(multiply(4, 2)); //

==========================================

4.Immediately Invoked Function Expression (IIFE)

A function that runs immediately after being defined.

(function() {

console.log("This runs immediately!");

})();

Which One to Use?

For simple calculations: Use arrow functions.

For reusable functions: Use named functions.

For one-time execution: Use IIFE.

For asynchronous tasks: Use callback functions or promises.

For handling collections: Use higher-order functions like map(), filter(), reduce().

For complex problems like tree traversal: Use recursion

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Type** | **Pros** | **Cons** | **Best For** |
| **Named Function** | Easy to debug, reusable, supports recursion | Longer syntax | General-purpose, reusable functions |
| **Anonymous Function** | Shorter, flexible | Harder to debug (no name in error stack) | One-time use, inline functions |
| **Arrow Function** | Concise, this is lexically bound | Not suitable for methods (this issues) | Simple functions, callbacks, event handlers |
| **IIFE (Immediately Invoked Function Expression)** | Runs immediately, avoids global scope pollution | Hard to reuse | Initialization code, self-executing logic |
| **Higher-Order Function** | Improves reusability, follows functional programming principles | Can be complex to understand | Functional programming, callbacks, array methods like map(), filter() |
| **Callback Function** | Useful for async operations (like setTimeout) | Can lead to callback hell | Asynchronous programming, event handling |
| **Recursive Function** | Solves problems like factorial, tree traversal | Can cause stack overflow if not properly handled | Problems involving recursion, like tree structures |