PROSENSIA PVT LTD
M FAROOQ
FUNCTION

Introduction

Functions are reusable blocks of code that perform specific tasks. In JavaScript, functions help organize and modularize code, making it easier to maintain. Understanding scopes ensures variables are accessed where they are intended.

---

1. Regular Functions

Regular functions are declared using the `function` keyword.

Syntax:

```
javascript
CopyEdit
function greet(name) {
    return 'Hello ' + name;
}
console.log(greet('Ali')); // Output: Hello Ali
```

Regular functions support **hoisting**, meaning they can be called before their declaration in the code.

---

2. Arrow Functions

Arrow functions provide a shorter syntax and do **not** bind their own `this`.

Syntax:

```
javascript
CopyEdit
const greet = (name) => {
    return 'Hello ' + name;
};
```

Arrow functions are **not hoisted** and are ideal for shorter functions or callbacks.

They're especially useful in array methods like `.map()`, `.filter()`, etc.

3. Function Scope

Scope determines the accessibility of variables.

Global Scope: Accessible anywhere in the program.

Function (Local) Scope: Variables defined inside a function are only accessible within it.

Block Scope: Introduced with `let` and `const`, applies within `{}` blocks like loops or conditionals.

Example:

```
javascript
CopyEdit
function testScope() {
    let local = 'I am local';
    console.log(local); // OK
}
testScope();// console.log(local); // ✕ Error: local is not defined
```

Summary

JavaScript supports both **regular** and **arrow** functions, each suited to different scenarios.

Understanding function **scope** ensures that variables behave predictably and avoids bugs caused by unintended access.

Mastery of functions and scopes leads to modular, readable, and maintainable JavaScript code.