**Understanding Scope :**

function outer() {

let outerVar = 10;

function inner() {

let innerVar = 20;

console.log(outerVar + innerVar);

}

inner();

}

outer();

**Arrow Function Syntax:**

let add = (a, b) => a + b;

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

**Function Hoisting:**

console.log(test());

function test() {

return "Hoisted!";

}

**Nested Function Return:**

function outer() {

function inner() {

return "Hello from inner";

}

return inner;

}

let fn = outer();

console.log(fn());

**Arrow Function and arguments Object:**

**​​**function test() {

let arrow = () => {

console.log(arguments[0]);

};

arrow(1, 2, 3);

}

test(4, 5, 6);

**Default Parameters:**

function greet(name = "Guest") {

return `Hello, ${name}!`;

}

console.log(greet());

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

**IIFE (Immediately Invoked Function Expression):**

(function() {

console.log("IIFE executed!");

})();

**Nested Function Closure:**

function counter() {

let count = 0;

return function() {

count++;

return count;

};

}

let count1 = counter();

console.log(count1());

console.log(count1());

let count2 = counter();

console.log(count2());

console.log(count1());

**Arrow Function Returning Object (without braces):**

let getObject = () => ({ name: "John" });

console.log(getObject());

**Function Expressions:**

let square = function(x) {

return x \* x;

};

console.log(square(5));

**Arrow Function Without Parameters:**

let greet = () => "Hello!";

console.log(greet());

**Function as a Return Value:**

function createMultiplier(factor) {

return function(x) {

return x \* factor;

};

}

let double = createMultiplier(2);

console.log(double(5));

**Function Arguments and Parameters:**

function sum(a, b) {

return a + b;

}

console.log(sum(5, 10));

console.log(sum(5));

console.log(sum());

**Function Overloading:**

function foo(x, y) {

if (y === undefined) {

y = x;

}

return x + y;

}

console.log(foo(5));

console.log(foo(5, 10));

**Function Declaration and Function Expression Difference:**

console.log(foo());

function foo() {

return "Function Declaration";

}

console.log(bar());

var bar = function() {

return "Function Expression";

};

**Closures in Nested Functions:**

function makeCounter() {

let count = 0;

return function() {

count++;

return count;

};

}

let counter1 = makeCounter();

let counter2 = makeCounter();

console.log(counter1());

console.log(counter1());

console.log(counter2());

console.log(counter2());