JavaScript Fundamentals: ES6+ Features, Functions, Arrays & Objects

PART 1: JavaScript Fundamentals & ES6+ Variables

1. Overview of JavaScript & ES6+
JavaScript is a high-level, interpreted programming language core to web development, running in browsers and servers (Node.js). ECMAScript 2015 (ES6) and subsequent modern versions introduced powerful features that make JavaScript code cleaner, more modular, and easier to maintain.

2. Variable Declarations: var, let, and const
- var: Function-scoped, can be redeclared and reassigned, and is hoisted with an initial value of undefined, making it prone to unexpected bugs.
- let: Block-scoped, can be reassigned but not redeclared within the same scope. It is hoisted but not initialized (results in a Temporal Dead Zone).
- const: Block-scoped, cannot be reassigned or redeclared. Must be initialized upon declaration (though object properties or array items inside a const can still be mutated).

3. Operators and Expressions
- Arithmetic Operators: +, -, *, /, %, ** (exponentiation).
- Comparison Operators: == (loose equality with type coercion), === (strict equality matching value and type), !=, !==, >, <, >=, <=.
- Logical Operators: && (AND), || (OR), ?? (Nullish coalescing - returns right side only if left is null/undefined), ?. (Optional chaining).
- Spread (...) and Rest (...) Operators (ES6+): Spread unpacks arrays/objects; rest collects remaining arguments into an array.

4. Control Structures
- Conditional Statements: if, else if, else, and switch-case.
- Looping Constructs: for, while, do-while, for...of (iterates over iterable values like arrays), and for...in (iterates over object keys).

PART 2: Functions, Arrow Functions, and Closures

1. Function Declarations vs Function Expressions
- Function Declarations are hoisted entirely, allowing them to be called before they are defined in code.
- Function Expressions assign an anonymous or named function to a variable and are not hoisted.

2. Arrow Functions (ES6+)
Arrow functions provide a concise syntax using the fat arrow (=>). They do not have their own 'this' context, inheriting 'this' lexically from their surrounding scope, making them ideal for callbacks.

3. Closures
A closure is a function bundled together with references to its lexical environment. It allows an inner function to access an outer function's variables even after the outer function has finished executing, enabling data privacy and state encapsulation.

PART 3: Arrays and Objects

1. Arrays and Advanced Methods
Arrays are ordered collections of data. ES6+ provides powerful functional array methods:
- map(): Creates a new array populated with the results of calling a provided function on every element.
- filter(): Creates a shallow copy of a portion of a given array filtered down to elements passing the test.
- reduce(): Executes a reducer function on each element, resulting in a single output value.

2. Objects and Modern Enhancements
Objects store collections of key-value pairs (properties and methods). ES6 introduced shorthand property names, concise method syntax, and computed property names, alongside Object.keys(), Object.values(), and Object.entries().

PART 4: Comprehensive JavaScript Program

Program: ES6+ Features, Arrow Functions, Closures, Arrays & Objects

```
// 1. Variables and Arrow Functions
const greetUser = (name = 'Guest') => `Welcome, ${name}!`;
console.log(greetUser('Asha'));

// 2. Closure Example (Data Privacy)
function createCounter() {
  let count = 0;
  return {
    increment: () => ++count,
    getCount: () => count
  };
}
const counter = createCounter();
counter.increment();
console.log(`Current Count: ${counter.getCount()}`);

// 3. Arrays and ES6+ Methods (map, filter, reduce)
const products = [
  { id: 1, name: 'Laptop', price: 1200, category: 'Electronics' },
  { id: 2, name: 'Mouse', price: 25, category: 'Electronics' },
  { id: 3, name: 'Desk', price: 300, category: 'Furniture' }
];
```

```
const expensiveElectronics = products
   .filter(p => p.category === 'Electronics')
   .map(p => ({ ...p, price: p.price * 0.9 })); // 10% discount
console.log('Discounted Electronics:', expensiveElectronics);

const totalPrice = products.reduce((sum, p) => sum + p.price, 0);
console.log(`Total Inventory Value: $${totalPrice}`);

// 4. Object Destructuring
const user = { username: 'Ravi', role: 'Admin', location: 'Chennai' };
const { username, role } = user;
console.log(`${username} is logged in as ${role}.`);
```