🚀 Day 2: Functions & Arrow Functions

Master JavaScript Functions, Spread/Rest Operators & Interview Concepts

📝 Function Declaration

Syntax: Uses `function` keyword

```
function greet(name) {
  return `Hello, ${name}!`;
}
```

Key Points:

- ✅ Hoisted - can use before declaration
- ✅ Named function
- ✅ Good for main logic

📦 Function Expression

Syntax: Assigned to a variable

```
const multiply = function(a,
b) {
  return a * b;
}
```

Key Points:

- ❌ NOT hoisted
- ✅ Can be anonymous
- ✅ Good for callbacks

⚡ Arrow Function (ES6+)

Syntax: Modern concise syntax

```
const add = (a, b) => a + b;
```

Key Points:

- ❌ NOT hoisted
- ✅ Implicit return (single line)

- ✅ Lexical this

🔄 Arrow Function Variations

Single Parameter

```
const square = x => x * x;
```

Parentheses optional for single param

Multiple Parameters

```
const add = (a, b) => a +
b;
```

Parentheses required for multiple

No Parameters

```
const greet = () => "Hi!";
```

Empty parentheses required

Return Object

```
const user = () => ({ name:
"Ali" });
```

Wrap in parentheses to return object

📡 Spread Operator (...)

Used to unpack/expand elements from arrays or objects

1️⃣ Spreading Arrays

```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const combined = [...arr1, ...arr2];
// Result: [1, 2, 3, 4, 5, 6]
```

2️⃣ Spreading Objects

```
const user = { name: "Ali", age: 25 };
const updated = { ...user, city: "Karachi" };
// Result: { name: 'Ali', age: 25, city: 'Karachi' }
```

3️⃣ Spreading in Function Calls

```
function sum(a, b, c) { return a + b + c; }
const nums = [1, 2, 3];
sum(...nums); // Result: 6
```

🎯 Rest Parameter (...)

Used to collect/gather multiple arguments into an array

1️⃣ Rest in Function Parameters

```
function printNumbers(...args) {
  console.log(args); // Array of all arguments
}
printNumbers(1, 2, 3, 4, 5);
// Output: [1, 2, 3, 4, 5]
```

2️⃣ Rest with Other Parameters

```
function greet(greeting, ...names) {
  names.forEach(name => {
    console.log(`${greeting}, ${name}!`);
  });
}
greet("Hello", "Ali", "Sara", "Omar");
```

3️⃣ Rest in Array Destructuring

```
const [first, second, ...rest] = [10, 20, 30, 40, 50];
// first: 10, second: 20, rest: [30, 40, 50]
```

4️⃣ Rest in Object Destructuring

```
const { name, ...otherInfo } = { name: "Ali", age: 25, city: "Karachi"
};
// name: "Ali", otherInfo: { age: 25, city: 'Karachi' }
```

⚖️ Spread vs Rest: Key Differences

Aspect	Spread (...)	Rest (...)
Purpose	Unpacks/Expands elements	Collects/Gathers elements
Position	RIGHT side of assignment	LEFT side of assignment
Example	[...arr], {...obj}	(...args), [...rest]
Used In	Function calls, array/object literals	Function parameters, destructuring

Aspect	Spread (...)	Rest (...)
Result	Multiple individual items	Single array/object

💡 **Same Syntax, Different Purpose!**

The three dots (...) look the same, but context determines if it's spread (unpacking) or rest (collecting). Remember: Spread = spread out, Rest = get the rest!

✨ Advantages of Arrow Functions

1

Concise Syntax

Write less code with cleaner, more readable functions

2

Implicit Return

Single-line functions return automatically

3

Lexical This

Inherits 'this' from parent scope

4

Array Methods

Perfect for map, filter, reduce operations

5

No Arguments

Use rest parameter instead of arguments object

6

Callbacks

Great for event handlers and promises

🌍 Real-World Examples

📊 Example 1: Calculator with Spread

```
const calculate = (operation, ...nums) => {
  if (operation === 'sum') {
    return nums.reduce((acc, n) => acc + n, 0);
  }
};
calculate('sum', 1, 2, 3, 4, 5); // 15
```

🧮 Example 2: API Response Handler

```
const apiResponse = {
  status: 200,
  data: { name: "Ali", email: "ali@example.com" }
};

const { status, data: { name, email } } = apiResponse;
```

🎨 Example 3: Config Merging

```
const defaultConfig = { timeout: 5000, cache: false };
const userConfig = { timeout: 10000 };
const final = { ...defaultConfig, ...userConfig };
```

🔍 Example 4: Array Operations

```
const nums = [1, 2, 3, 4, 5];
const doubled = nums.map(n => n * 2); // [2, 4, 6, 8, 10]
const evens = nums.filter(n => n % 2 === 0); // [2, 4]
const sum = nums.reduce((acc, n) => acc + n, 0); // 15
```

💪 Practice Questions

Q1: Average Calculator with Rest Parameter

Create a function that accepts multiple numbers and returns their average

```
const average = (...nums) => {
  const sum = nums.reduce((acc, n) => acc + n, 0);
  return sum / nums.length;
};
average(10, 20, 30); // 20
```

Q2: Merge Objects

Create a function that merges two objects using spread

```
const mergeObjects = (obj1, obj2) => ({ ...obj1, ...obj2 });
mergeObjects({ a: 1 }, { b: 2 }); // { a: 1, b: 2 }
```

Q3: Split Array

Separate first element from rest using destructuring

```
const splitArray = (arr) => {
  const [first, ...rest] = arr;
  return { first, rest };
};
splitArray([1, 2, 3, 4, 5]);
```

Q4: Filter by Type

Filter items by their type using rest parameter

```
const filterByType = (type, ...items) => {
  return items.filter(item => typeof item === type);
};
filterByType('string', 1, 'hello', 2, 'world'); // ['hello',
'world']
```

✅ Master Day 2 Concepts and Ace Your JavaScript Interviews! 🎯