JavaType Script Guru - JavaScript practice questions

🔥 Nice, bro — love that energy! That's exactly how you master JavaScript — by *grinding through tons of questions* until it clicks deep down.
Below, I've crafted **a ton of structured practice questions** — starting from conceptual to coding — covering *Functions* and *Arrays & Objects* in detail.

You can treat these as **levels** — from warm-up to pro-mode. 🚀

🧠 SECTION 1 — FUNCTIONS IN JAVASCRIPT

🎯 Level 1: Conceptual Warm-Up

1. What is a function in JavaScript, and why are they called "first-class citizens"?
2. Explain the difference between **parameters** and **arguments** in JavaScript.
3. What's the difference between **function declaration** and **function expression**?
4. Define **hoisting**. How does function hoisting differ from variable hoisting?
5. Why do we say that "functions create scope"?
6. What's the difference between **global scope** and **function scope**?
7. Explain what a **closure** is, and give one real-life use case for it.

💻 Level 2: Hands-On Coding

1. Create a function `sum(a, b)` that returns the sum of two numbers.
   - Try calling it *before* and *after* its declaration — what happens? Why?
2. Write a function `multiply(...numbers)` that multiplies any number of arguments using **rest parameters**.
3. Write a function `greet({name, age})` that uses **destructured parameters**.
4. Implement a function `sayHello()` inside another function `outer()`.
   - Can `sayHello()` access variables defined in `outer()`?
5. Convert the following function into an **arrow function** and test if `this` behaves differently:

```
js
```

```
function showThis() {
  console.log(this);
}
```

6. Create an **IIFE** that prints `"I am immediately invoked!"`.
7. Write a **higher-order function** `repeat(fn, n)` that executes a given function `fn` exactly `n` times.
8. Create a **pure function** that doubles a number and another **impure function** that modifies a global variable.
9. Demonstrate **callback functions** using `setTimeout()`.
10. Write a **closure** function `counter()` that returns an object with `increment` and `getCount` methods.

🧩 Level 3: Deep Thinking

1. Why is a closure powerful in JavaScript?
2. In which cases is an arrow function *not suitable* to replace a classic function?
3. Explain why hoisting might cause unexpected results in large codebases.
4. Describe how **scope chain** works when you have nested functions.
5. Write an example showing that function declarations are hoisted *fully*, but variable assignments are *not*.

🧱 SECTION 2 — ARRAYS & OBJECTS IN JAVASCRIPT

🎯 Level 1: Conceptual Warm-Up

1. What are arrays and why are they called *objects* in JS?
2. How are arrays stored internally in JavaScript?
3. What's the difference between `for`, `for...of`, and `forEach()` when looping over an array?
4. Explain what `array destructuring` is.
5. What are the differences between `slice()`, `splice()`, and `split()`?
6. Why does `sort()` behave unexpectedly for numbers sometimes?

💻 Level 2: Hands-On Coding

1. Create an array `fruits = ['apple', 'banana', 'cherry']`.
   - Add `'mango'` at the end and `'orange'` at the beginning.
   - Remove `'banana'` from the middle.
2. Use `map()` to square all numbers in an array `[2, 4, 6, 8]`.
3. Use `filter()` to return only even numbers from `[1, 2, 3, 4, 5, 6]`.
4. Use `reduce()` to calculate the total sum of `[10, 20, 30]`.
5. Use the **spread operator** to merge two arrays.
6. Use **array destructuring** to extract the first two elements of `[1, 2, 3, 4]`.
7. Create an array of objects representing students and use `find()` to get one by name.
8. Reverse an array without using `.reverse()` (hint: use a loop).
9. Convert `['A', 'B', 'C']` to `'A-B-C'` using `join()`.
10. Use `some()` to check if any number in `[5, 10, 15, 20]` is greater than 12.

🧱 Level 3: Objects Deep Dive

1. Create an object `user` with `name`, `age`, and `email`.
   - Add a new property `isAdmin` dynamically.
   - Delete the `email` property.
2. Create a nested object and access a deeply nested property safely.
3. Loop over object keys and values using `for...in`.
4. Convert an object to an array of key-value pairs using `Object.entries()`.
5. Merge two objects using the spread operator.
6. Create an array of objects and sort them by a property (e.g., `age`).
7. Demonstrate that arrays are objects by adding a custom property to an array.
8. Write a function that counts occurrences of each character in a string using an object.
9. Explain what happens when you copy an object with `=` versus with `structuredClone()` or `JSON.parse(JSON.stringify())`.

⚙️ Level 4: Combine All (Pro-Level)

1. Create a function `analyzeScores(students)` that takes an array of student objects:

```
js
```

```
[
  { name: 'Alice', score: 85 },
  { name: 'Bob', score: 92 },
  { name: 'Charlie', score: 78 }
]
```

and returns:

```
js

{
  average: 85,
  highest: 'Bob',
  lowest: 'Charlie'
}
```

2. Implement a **function factory** using closures — e.g. `createMultiplier(factor)` that returns a new function to multiply numbers by `factor`.
3. Build a simple **to-do list manager** using an array of objects. Include `addTodo`, `deleteTodo`, and `getTodos` functions.
4. Combine **map**, **filter**, and **reduce** to transform and summarize data in one pipeline.
5. Write an IIFE that creates a private object and exposes only a few methods — simulating a simple module.

Would you like me to:

- 🧩 Generate **practice exercises with solutions** (step-by-step explanation)?
- ⚔️ Or **convert these into a 30-day coding challenge** (daily tasks for mastery)?

That way I can structure it to match your grind plan.