Loops + Functions + Switch — TODOs

TODO-1: Implement runOperation(arr, op) (Functions + Switch + Loops)

Problem Statement
Complete the function runOperation(arr, op) so that it returns a single result based on the selected operation.
The parameter op is a string chosen from the dropdown: "sum", "product", "max", or "count-even".
Use a switch statement to select the operation and loops (for or while) to compute results.

What to implement
Inside runOperation(arr, op):
- Use switch (op) { ... }.
- For each case:
- sum: return the sum of all numbers in arr.
- product: return the product of all numbers in arr.
- If arr is empty, return 0.
- max: return the largest number in arr.
- If arr is empty, return "N/A".
- count-even: return how many numbers in arr are even (i.e., divisible by 2).
- For any unknown operation, return "Unknown op".

Constraints & Rules
- You must use a switch.
- Use at least one loop for the computations (no reduce for this task).
- Do not modify HTML/CSS; edit only app.js.

Examples
- arr = [5, 2, 7, 2, 8], op = "sum" -> 24
- arr = [5, 2, 7, 2, 8], op = "product" -> 1120
- arr = [5, 2, 7, 2, 8], op = "max" -> 8
- arr = [5, 2, 7, 2, 8], op = "count-even" -> 3
- arr = [], op = "max" -> "N/A"
- arr = [], op = "product" -> 0
- arr = [1, 3, 5], op = "count-even" -> 0

Hints
- Initialize sum with 0, product with 1, and max with the first element (if array not empty).
- Even check: n % 2 === 0.

Acceptance Criteria
- Uses switch with all required cases.
- Correct results for the provided examples.
- No crashes for empty arrays.

TODO-2: Implement summarizeEvenOdd(arr) (Loops + Function Return)

Problem Statement
Complete the function summarizeEvenOdd(arr) to count even and odd numbers in the provided array and return an object with two properties: { even: , odd: }.

What to implement
Inside summarizeEvenOdd(arr):
- Use a loop to traverse arr.
- For each number:
- If even -> increment even
- Else -> increment odd
- Return the final object: { even, odd }.

Constraints & Rules
- Must use a loop (for or while).
- Do not modify function parameters or the return shape.
- No array helpers (filter, reduce) for counting—explicit loop required.

Examples
- arr = [5, 2, 7, 2, 8] -> { even: 3, odd: 2 }
- arr = [1, 3, 5] -> { even: 0, odd: 3 }
- arr = [] -> { even: 0, odd: 0 }

Hints
- Initialize: let even = 0, odd = 0;
- Even check: n % 2 === 0.
- Return exactly { even, odd }.

Acceptance Criteria
- Returns an object with both keys present.
- Correct counts for the examples above.
- Works for empty arrays without errors.