Problem 2626: Array Reduce Transformation

Problem Information

Difficulty: Easy
Acceptance Rate: 85.45%
Paid Only: No

Problem Description

Given an integer array `nums`, a reducer function `fn`, and an initial value `init`, return the final result obtained by executing the `fn` function on each element of the array, sequentially, passing in the return value from the calculation on the preceding element.

This result is achieved through the following operations: `val = fn(init, nums[0]), val = fn(val, nums[1]), val = fn(val, nums[2]), ...` until every element in the array has been processed. The ultimate value of `val` is then returned.

If the length of the array is 0, the function should return `init`.

Please solve it without using the built-in `Array.reduce` method.

Example 1:

Input: nums = [1,2,3,4] fn = function sum(accum, curr) { return accum + curr; } init = 0 **Output:** 10 **Explanation:** initially, the value is init=0. (0) + nums[0] = 1 (1) + nums[1] = 3 (3) + nums[2] = 6 (6) + nums[3] = 10 The final answer is 10.

Example 2:

Input: nums = [1,2,3,4] fn = function sum(accum, curr) { return accum + curr * curr; } init = 100 **Output:** 130 **Explanation:** initially, the value is init=100. (100) + nums[0] * nums[0] = 101 (101) + nums[1] * nums[1] = 105 (105) + nums[2] * nums[2] = 114 (114) + nums[3] * nums[3] = 130 The final answer is 130.

Example 3:

Input: nums = [] fn = function sum(accum, curr) { return 0; } init = 25 **Output:** 25
Explanation: For empty arrays, the answer is always init.

Constraints:

* `0 <= nums.length <= 1000` * `0 <= nums[i] <= 1000` * `0 <= init <= 1000`

Code Snippets

JavaScript:

```
/**
* @param {number[]} nums
* @param {Function} fn
* @param {number} init
* @return {number}
*/
var reduce = function(nums, fn, init) {

};
```

TypeScript:

```
type Fn = (accum: number, curr: number) => number

function reduce(nums: number[], fn: Fn, init: number): number {

};
```