

I am writing tests for a csv parser that could either return an array of schemas or 2-D array of strings (if the schemas aren't properly validated) Because there are two return types, however, it is hard to test that the schema methods work. Any ideas?

TS basic-parser.test.ts

Here are some ideas to make your tests robust for a CSV parser that can return either an array of schemas or a 2-D array of strings:

1. **Type Guard Function:**

Write a helper function to distinguish between schema objects and string arrays. For example:

```
function isSchemaRow(row: any
```

@workspace /explain parseCSV works with schemas class

Error:

```
expect(received).toBe(expected)  
 // Object.is equality
```

Expected: true

Received: false



Workspace

> Used 1 reference

This error means that

`Array.isArray(results)` returned `false`, so `results` is not an array.

Possible causes:

- `parseCSV` may return an object (like an error) instead of an array if schema validation fails.
- The schema or CSV file may not

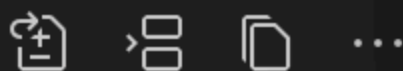


I do not expect an error from undefined input though, I expect a 2-D array of strings

If you expect `parseCSV` to return a 2-D array of strings when the schema is `undefined`, you should write a test that:

- Passes `undefined` as the schema argument.
- Asserts that the result is an array of arrays of strings.

Example:



```
test("parseCSV returns 2-D array of strings", () => {
  const results = await parseCSV(undefined, schema);
  expect(Array.isArray(results)).toBe(true);
  for (const row of results) {
    expect(Array.isArray(row)).toBe(true);
    for (const cell of row) {
      expect(typeof cell).toBe('string');
    }
  }
});
```