const externalFuncsWithDefs = Object.getOwnPropertyNames(window)

.filter(prop => {

const fn = window[prop];

return typeof fn === 'function' &&

fn.toString().indexOf('[native code]') === -1;

})

.map(name => ({

name: name,

function: window[name]

}));

console.log(externalFuncsWithDefs);

If you're looking to implement a simple input-output logging system for functions without using a database, you can create a logging mechanism that captures the inputs, outputs, and function names in memory. This can be useful for debugging, monitoring, or understanding how your functions are being used.

**Step 1: Create a Logging Function**

You can create a generic logging function that will wrap any target function, keeping track of its inputs and outputs.

Here’s how you can do it:

**const** logs **=** []; *// Array to store function logs*

**function** logFunction(func) {

**return** **function**(...inputs) {

**const** output **=** func(...inputs); *// Call the original function*

**const** logEntry **=** {

functionName: func.name **||** 'anonymous',

inputs: inputs,

output: output,

timestamp: **new** Date().toISOString() *// Optional: save the timestamp*

};

logs.push(logEntry); *// Store the log entry*

console.log('Function executed:', logEntry); *// Log to console*

**return** output; *// Return the original output*

};

}

**Step 2: Example Usage**

You can use the logFunction to wrap any function you want to monitor. Here’s an example with a simple addition and multiplication function:

*// Example functions*

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

**const** multiply **=** (a, b) => a **\*** b;

*// Wrap functions with logging*

**const** loggedAdd **=** logFunction(add);

**const** loggedMultiply **=** logFunction(multiply);

*// Use the logged functions*

loggedAdd(5, 10); *// Logs: Function executed: { functionName: 'add', inputs: [5, 10], output: 15, timestamp: '...' }*

loggedMultiply(3, 4); *// Logs: Function executed: { functionName: 'multiply', inputs: [3, 4], output: 12, timestamp: '...' }*

*// Check the logs array*

console.log(logs);

**Explanation**

1. **Logging Structure**:
   * The logs array is used to store entries for each function call.
   * Each log entry contains:
     + functionName: The name of the function called (if available).
     + inputs: An array of the inputs provided to the function.
     + output: The result returned by the function.
     + timestamp: A timestamp indicating when the function was called.
2. **Log Function**:
   * The logFunction takes a function as an argument and returns a new function. This new function calls the original function, captures the inputs and outputs, and stores this information in the logs array.
   * It also logs the details to the console for immediate feedback.
3. **Example Usage**:
   * In the example, we wrap the add and multiply functions. When these functions are called, their inputs and outputs are logged, allowing you to track their usage easily.