Memory Management and DOM Optimization

Steve Buchanan

DevOps Architect

@buchatech | www.buchatech.com

Overview

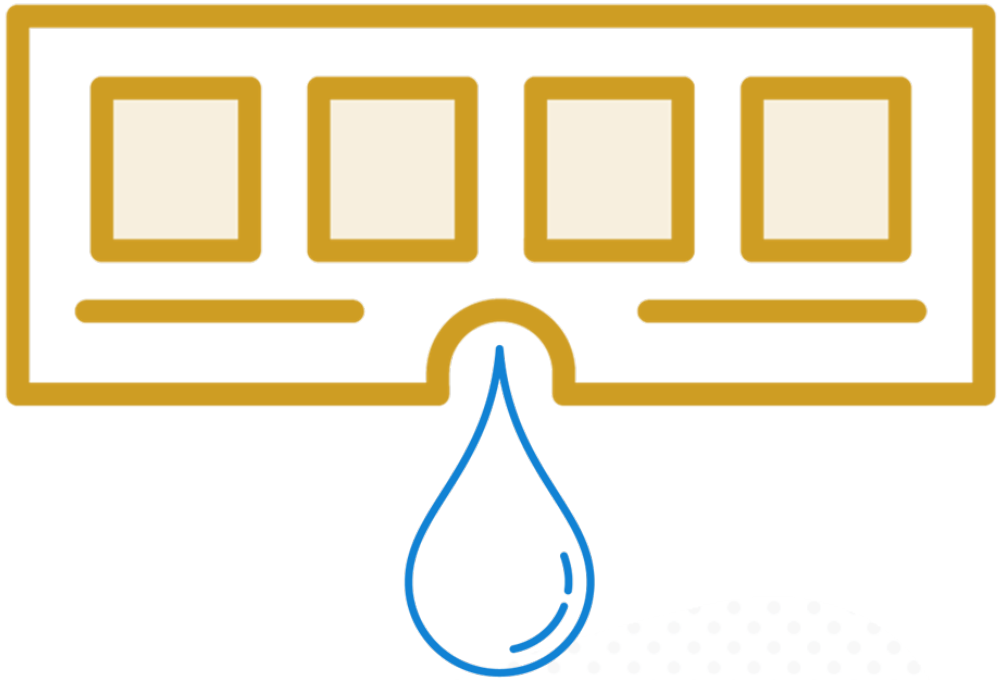
Handling Memory Leaks within a JavaScript Application

Optimizing Document Object Model (DOM) in JavaScript

Reviewing Variable Scope for Performance Optimization

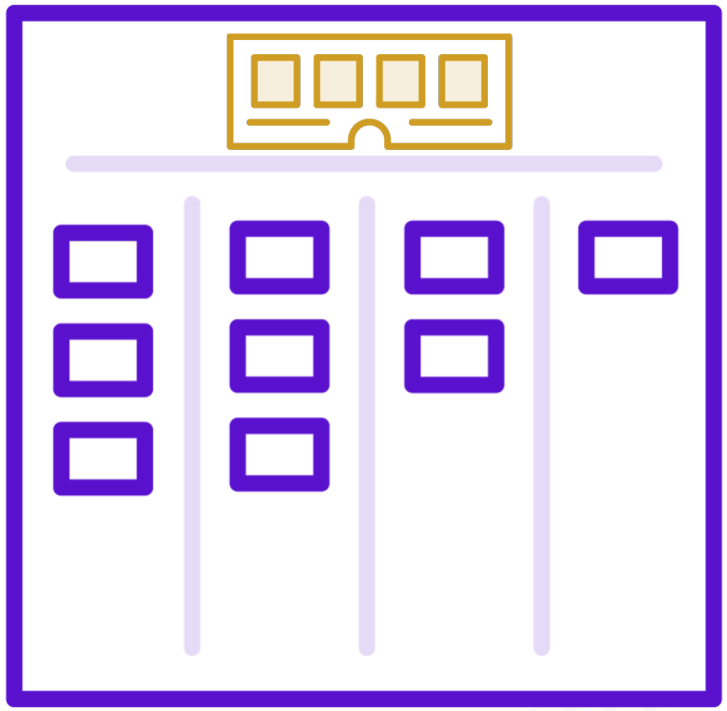
Handling Memory Leaks within a JavaScript Application

Understanding a Memory Leak



A memory leak is when memory that is not required by an app anymore for a variety of reasons is not returned to the OS or the pool of free memory.

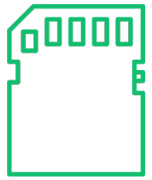
Memory Management in JavaScript



JavaScript is a garbage collected language.

Garbage collected languages help developers manage memory by periodically checking which previously allocated pieces of memory can still be "reached" from other parts of the application.

Three Common JavaScript Memory Leaks



Forgotten timers or callbacks

Accidental global variables

Out of DOM references

Handling Memory Leaks within a JavaScript Application

Using tools such as Chrome DevTools or Monitoring>>APM tools like Datadog

Use Profiling or Heap Snapshots to diagnose & pinpoint the exact issue & root cause of the memory leak

Remediate the leak by fixing the code, optimizing code/framework

Demo

Demo: Detecting a Memory Leak

Optimizing Document Object Model (DOM) in JavaScript

Review: What is DOM?

The Document Object Model (DOM) is the way web developers manipulate the structure & content of a webpage using JavaScript.

How Can DOM Impact Performance

When we interact with the DOM using JavaScript, we trigger a series of actions that can be resource-intensive.

For instance, every time we add or remove an element from the DOM, the browser recalculates the layout of the page, leading to delays in rendering the content.

Optimizing DOM in JavaScript

Three ways to optimize DOM:

Minimizing searching for DOM nodes

Cache DOM references

Every time you query the DOM for an element, the browser has to search through the document to find it, which can be slow. To avoid this, you can cache the reference to the element and reuse it whenever you need to interact with it.

Reduce DOM changes

Batch DOM Interactions

Grouping multiple updates to the Document Object Model (DOM) into a single operation rather than making individual changes one by one.

Reducing DOM access, size, and elements

Event Delegation

Instead of attaching event listeners to each child element individually, you can attach them to a parent element and delegate the events to the child elements.

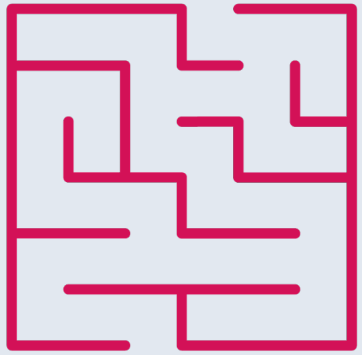
Reviewing Variable Scope for Performance Optimization

Optimization of Variable Scopes for Better Performance

Review variable scope

Ensure variables are not elevated beyond where they should be

Review Variable Scope



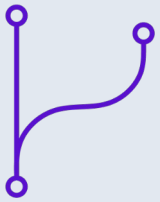
Understanding how and where variables are accessible within your code.

Main types of scopes in JavaScript:

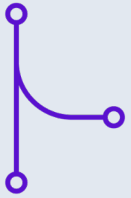
- Global Scope:
  - Variables declared outside any function or block have global scope and are accessible from anywhere in the code.
- Function Scope:
  - Variables declared within a function using var are only accessible within that function.
- Block Scope:
  - Variables declared within a block (e.g., inside {} braces) using let or const are only accessible within that block.

Ensure Variables Are Not Elevated beyond Where They Should Be

Best Practices for Scope Management



Minimize the use of global variables to avoid conflicts and unintended side effects.



Encapsulate code within functions or blocks to create local scopes, making the code more modular & maintainable.

Summary

In this module we covered:

- Handling Memory Leaks within a JavaScript Application
- Optimizing Document Object Model (DOM) in JavaScript
- Reviewing Variable Scope for Performance Optimization

Why this is important:?

- At some point you will run into the issue of a memory leak. Know what types there are and how to detect it will come in handy. Also knowing how to optimize DOM and best practices for variable scopes will serve you well when developing and running JavaScript web apps.