JavaScript Execution Context - Lecture Notes

How JavaScript Works & Execution Context - Akshay Saini (Namaste JavaScript Ep.1)

Understanding the internal workings of JavaScript is crucial for becoming a proficient developer. This lecture delves into the core concepts of how JavaScript operates behind the scenes, focusing on the Execution Context.

JavaScript Engine Components

---

1. Memory Heap: Stores objects and functions in memory.
2. Call Stack: Manages function execution order.

Execution Context

An Execution Context is an environment where JavaScript code is executed.

Components:

1. Memory Component: Stores variables/functions (initialized as undefined).
2. Code Component: Executes code line-by-line.

Phases of Execution Context

---

1. Creation Phase: Memory is allocated (variables as undefined, functions by reference).
2. Execution Phase: Actual code runs; values assigned and functions invoked.

Global Execution Context

- Created when JavaScript starts.
- Represents the outermost scope.
- 'this' refers to global object (e.g., window in browsers).

JavaScript Execution Context - Lecture Notes

Function Execution Context

- Created when a function is called.
- Maintains its own memory and code execution.

Call Stack

- Tracks function call order.
- Functions are "pushed" when called and "popped" when completed.

Synchronous and Single-Threaded Nature

- JS is synchronous and single-threaded.
- Executes one command at a time via the call stack.

Summary

- Execution Context = Memory + Code Environment.
- Global Context is default.
- New context created per function call.
- Call Stack ensures orderly execution.

Watch full video: https://www.youtube.com/watch?v=ZvbzSrg0afE