JavaScript Execution Context: The Core Fundamentals

JavaScript is a lightweight, interpreted programming language used to create dynamic web pages. It runs on the client-side and supports object-oriented, functional, and imperative programming styles. JavaScript allows interaction with HTML & CSS, event handling, and API integration.

Understanding JavaScript Execution

Execution Context

JavaScript code execution happens inside the **execution context**, which acts as a container for JavaScript code.

Key fundamentals of execution context:

- It consists of two components: **memory** and **code** components.
- The memory component stores variables and functions as key-value pairs.
- The code component executes the code one line at a time.

Components of Execution Context

Execution context can be visualized as follows:

Memory Component (Variable Environment)

- Stores variables and function declarations as key-value pairs.

Code Component (Thread of Execution)

- Executes JavaScript code line by line.

Example Representation:

```
Memory               | Code (Thread of Execution)
---------------------|--------------------------
key:value pair       | line of code
  a  : 10            | ------------
  fn : {..}          | line of code
```

JavaScript: Synchronous and Single-Threaded

JavaScript is a **synchronous, single-threaded** language, meaning:

- It can only execute **one command at a time**.
- Commands are executed **line by line** in a specific order.

Example: Execution Context Creation

Consider the following JavaScript code:

```
var a = 10;
function square() {
  var result = 20;
  return 20;
}
```

Execution Process

1. **Memory Allocation Phase:**
   - `a` is stored in memory with the value `undefined`.
   - `square` function is stored as a reference in memory.
2. **Execution Phase:**
   - `a` is assigned the value `10`.
   - When `square()` is called, a **new execution context** is created.

Call Stack Representation

When the function `square` is invoked, a new execution context is created and pushed onto the **Call Stack**

```
| square   |  <- Function Execution Context (FEC)
|----------|
| GEC      |  <- Global Execution Context (GEC)
------------
```

Summary

- JavaScript executes code within an **Execution Context**, which consists of **Memory** and **Code Components**.
- The **Memory Component** (Variable Environment) stores variables and function declarations.
- The **Code Component** (Thread of Execution) runs the code line by line.
- JavaScript is a **synchronous, single-threaded** language, executing one command at a time in a specific order.
- The **Call Stack** manages execution contexts, pushing and popping function execution contexts as they are called and completed.

Additional Resources

For a more detailed explanation, refer to Akshay Saini's JavaScript YouTube series:
Understanding JavaScript Execution