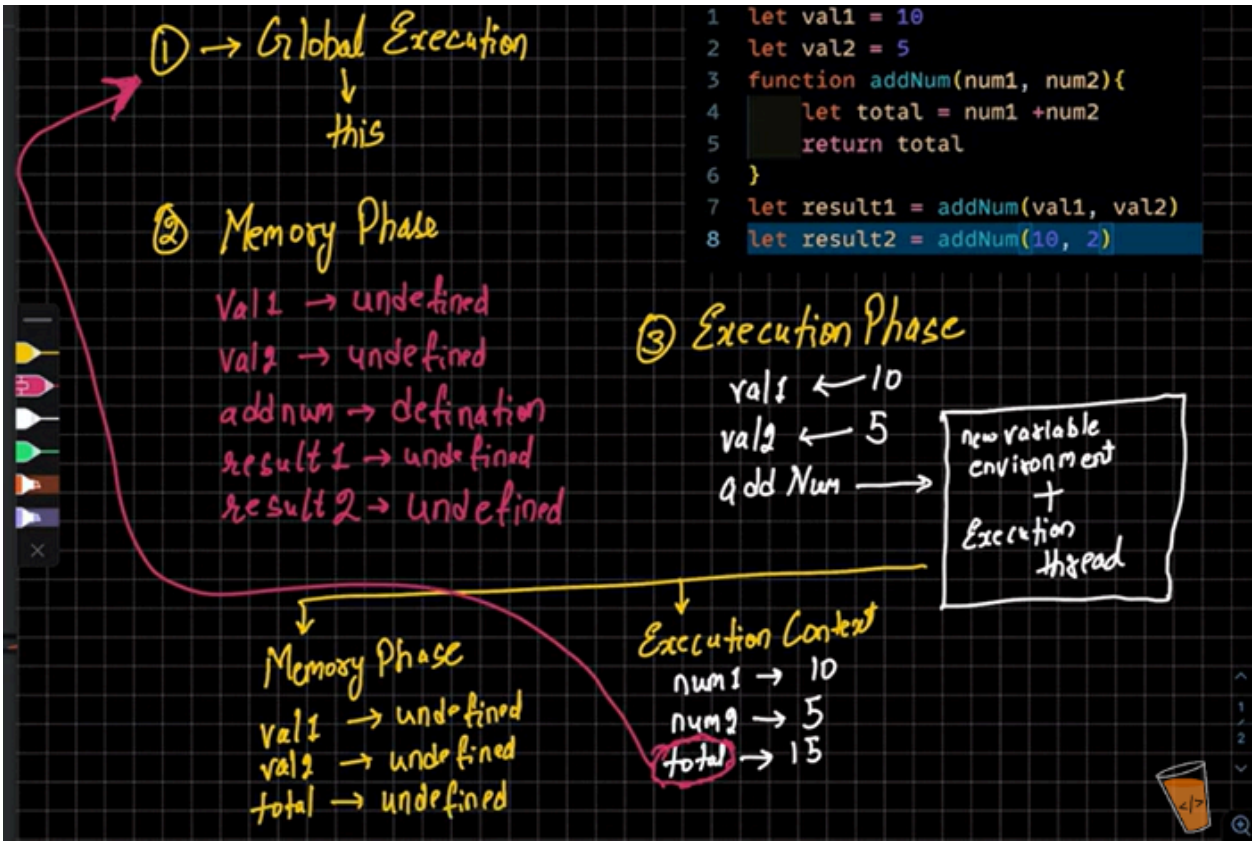
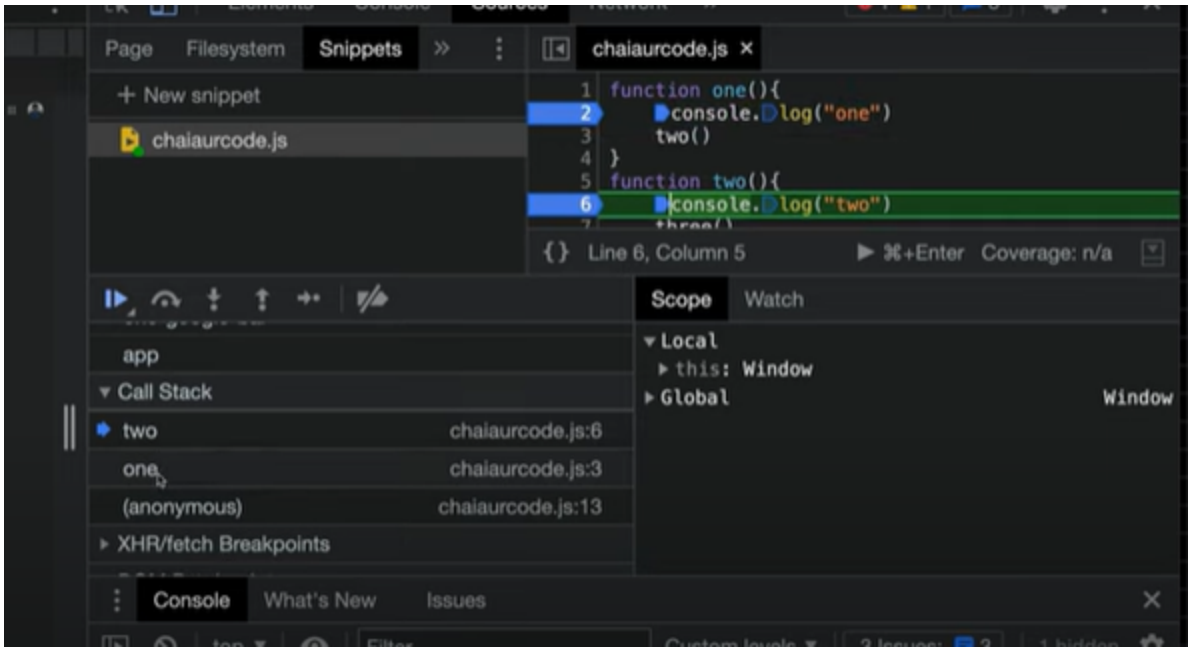
Code Execution & Call Stack

Execution Context



Call Stack



setTimeout()

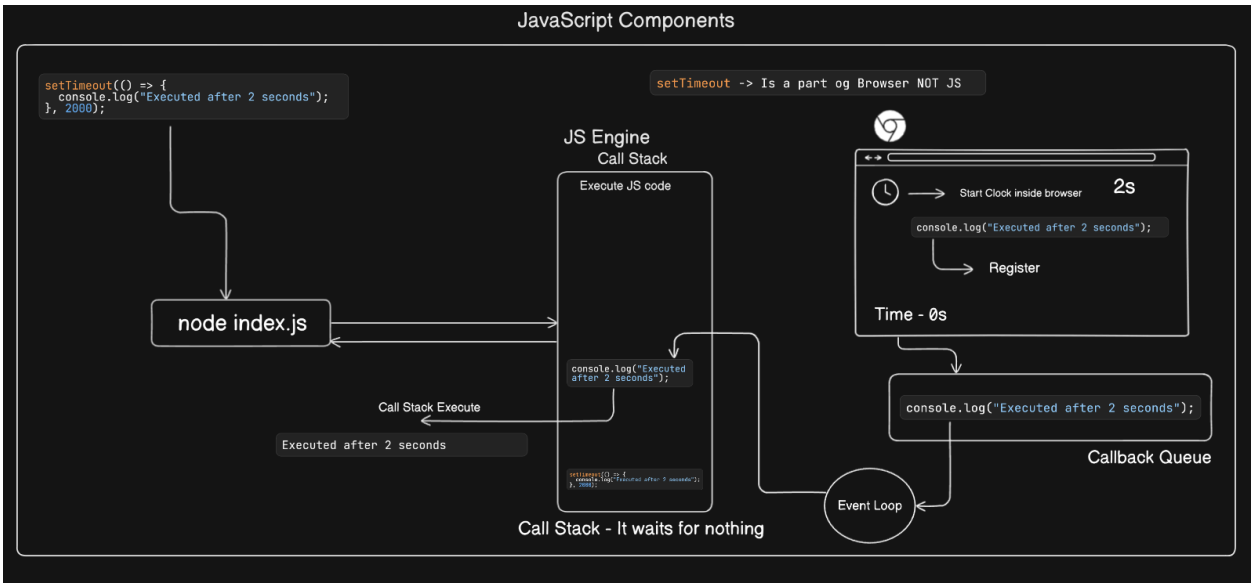
- `setTimeout` is a JavaScript function that executes a piece of code or a function after a specified delay (in milliseconds).
- It is commonly used for scheduling tasks, creating delays, and handling asynchronous operations.

Syntax

```
setTimeout(function, delay, param1, param2, ...);
```

Parameters:

1. `function` - The function to execute after the delay.
2. `delay` - The time in milliseconds to wait before executing the function.
3. `param1, param2, ...` *(optional)* - Arguments passed to the function when executed.



Examples

1. Basic

```
setTimeout(() => {
    console.log("Hello after 2 seconds");
}, 2000);

// Prints "Hello after 2 seconds" after a delay of 2000ms (2 seconds).
```

2. Using Named Function

```
function greet(name) {
    console.log(`Hello, ${name}!`);
}

setTimeout(greet, 3000, "John");

//  Executes greet("John") after 3 seconds.
```

3. Cancelling `setTimeout` with `clearTimeout`

```
let timerId = setTimeout(() => {
  console.log("This will not run");
}, 5000);

clearTimeout(timerId); // Cancels the timeout before execution

// console.log(timerId);
// timerId(); // TypeError: timerId is not a function
```

```
console.log("Hi");

setTimeout(() => console.log("Hello after 0 sec"), 0);
Promise.resolve().then(() => console.log("Promise resolved"));
setTimeout(() => console.log("Hello after 0 sec"), 0);

console.log("Bye");
```