

JAVASCRIPT EVENT LOOP



WHAT IS THE EVENT LOOP?

THE HEART OF JAVASCRIPT'S
NON-BLOCKING ARCHITECTURE.

HANDLES ASYNCHRONOUS TASKS
LIKE TIMERS, NETWORK REQUESTS,
AND EVENT LISTENERS.

ENABLES SMOOTH USER
EXPERIENCES.

JAVASCRIPT IS SINGLE-THREADED

EXECUTES ONE TASK AT A TIME
ON THE MAIN THREAD.

ACHIEVES CONCURRENCY USING
THE EVENT LOOP.

VISUAL: A SINGLE PIPELINE WITH
TASKS BEING PROCESSED
SEQUENTIALLY.

KEY COMPONENTS

CALL STACK: EXECUTES FUNCTION CALLS SEQUENTIALLY.

WEB APIs: HANDLES ASYNC OPERATIONS (E.G., SETTIMEOUT, FETCH).

TASK QUEUE: QUEUES TASKS LIKE SETTIMEOUT CALLBACKS.

MICROTASK QUEUE: HANDLES HIGH-PRIORITY TASKS (E.G., PROMISES).

HOW IT WORKS

1ST STEP

PROCESSES TASKS FROM THE CALL STACK.

2ND STEP

DELEGATES ASYNC OPERATIONS TO WEB APIs.

3RD STEP

QUEUES COMPLETED ASYNC CALLBACKS IN TASK/MICROTASK QUEUES.

4TH STEP

EVENT LOOP ENSURES THE CALL STACK IS EMPTY BEFORE EXECUTING QUEUED TASKS.



EXAMPLE 01

```
console.log('Start');

setTimeout(() => {
  console.log('Timeout');
}, 0);

Promise.resolve().then(() => {
  console.log('Promise');
});

console.log('End');
```



OUTPUT

01. START

02. END

03. PROMISE

04. TIMEOUT



WHY?

01. SYNCHRONOUS TASKS RUN FIRST (START, END).
02. PROMISE GOES TO THE MICROTASK QUEUE.
03. TIMEOUT GOES TO THE TASK QUEUE.
04. EVENT LOOP PROCESSES MICROTASKS FIRST, THEN TASKS.



JEXAMPLE 02

```
const foo = () =>
  console.log("First");

const bar = () =>
  setTimeout(() => console.log("Second"),
500);
const baz = () =>
  console.log("Third");

bar();
foo();
baz();
```



OUTPUT

FIRST

THIRD

SECOND



WHY?

01. WE INVOKE BAR. BAR RETURNS A SETTIMEOUT FUNCTION.
02. THE CALLBACK WE PASSED TO SETTIMEOUT GETS ADDED TO THE WEB API, THE SETTIMEOUT FUNCTION AND BAR GET POPPED OFF THE CALLSTACK.



03. THE TIMER RUNS, IN THE MEANTIME FOO GETS INVOKED AND LOGS FIRST. FOO RETURNS (UNDEFINED), BAZ GETS INVOKED, AND THE CALLBACK GETS ADDED TO THE QUEUE.

04. BAZ LOGS THIRD. THE EVENT LOOP SEES THE CALLSTACK IS EMPTY AFTER BAZ RETURNED, AFTER WHICH THE CALLBACK GETS ADDED TO THE CALL STACK.

05. THE CALLBACK LOGS SECOND.



VISUALISATION

<http://latentflip.com/loupe>



<https://www.linkedin.com/in/hemantha-wijegunarathna>

WHY EVENT LOOP MATTERS

ENABLES NON-BLOCKING,
SMOOTH USER EXPERIENCES.

HANDLES ASYNC OPERATIONS
LIKE API CALLS EFFECTIVELY.

PRIORITIZES TASKS FOR BETTER
PERFORMANCE.

PREVENTS THE UI FROM
FREEZING.



CONCLUSION

WRITE EFFICIENT, RESPONSIVE APPS.

AVOID BLOCKING THE MAIN THREAD.

DEBUG ASYNC BEHAVIOR EFFECTIVELY.

