JavaScript Event Loop

Back to JS page

Table of Contents

- JavaScript Event Loop
  - JavaScript Executes One Task at a Time
    - Example 1
  - Example: setTimeout()
    - Example 2
  - Example: Promise
    - Example 3
  - Task Queue and Microtask Queue
    - Example 4
  - Long Tasks Block JavaScript
    - Example 5
  - Document
  - Reference

JavaScript Executes One Task at a Time

JavaScript is single-threaded. It can only execute one task at a time.

However, the browser has Web APIs (like setTimeout, fetch, DOM events) that run in the background.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Event Loop</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>JavaScript Executes One Task at a Time</h4>
<p id="demo"></p>

<script>
function first() {
  document.getElementById("demo").innerHTML += "1. First function<br>";
}

function second() {
  document.getElementById("demo").innerHTML += "2. Second function<br>";
}

function third() {
  document.getElementById("demo").innerHTML += "3. Third function<br>";
}

// Functions execute in order, one at a time
first();
second();
third();
</script>

</body>
</html>
```

Example 1

Result View Example

Example: setTimeout()

When `setTimeout()` is called, the browser handles the timer. When the timer expires, the callback is moved to the **Task Queue**. The Event Loop picks it up when the call stack is empty:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Event Loop</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Example: setTimeout()</h4>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML += "1. Start<br>";

setTimeout(function() {
  document.getElementById("demo").innerHTML += "3. setTimeout callback<br>";
}, 0);

document.getElementById("demo").innerHTML += "2. End<br>";
</script>

</body>
</html>
```

Example 2

Result View Example

Example: Promise

Promises use the **Microtask Queue** which has higher priority than the Task Queue. Microtasks execute before the next task:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Event Loop</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Example: Promise</h4>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML += "1. Start<br>";

// Promise goes to Microtask Queue
Promise.resolve().then(function() {
  document.getElementById("demo").innerHTML += "3. Promise callback<br>";
});

// setTimeout goes to Task Queue
setTimeout(function() {
  document.getElementById("demo").innerHTML += "4. setTimeout callback<br>";
}, 0);

document.getElementById("demo").innerHTML += "2. End<br>";
</script>

</body>
</html>
```

Example 3

Result View Example

Task Queue and Microtask Queue

The Event Loop processes all microtasks before moving to the next task. This means Promise callbacks execute before setTimeout callbacks:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Event Loop</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Task Queue and Microtask Queue</h4>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML += "1. Synchronous code<br>";

// Promise (Microtask Queue)
Promise.resolve().then(function() {
  document.getElementById("demo").innerHTML += "3. Promise (microtask)<br>";
});

// Promise (Microtask Queue)
Promise.resolve().then(function() {
  document.getElementById("demo").innerHTML += "4. Another promise (microtask)<br>";
});

// setTimeout (Task Queue)
setTimeout(function() {
  document.getElementById("demo").innerHTML += "5. setTimeout (task)<br>";
}, 0);

document.getElementById("demo").innerHTML += "2. End of synchronous code<br>";
</script>

</body>
</html>
```

Example 4

Result View Example

Long Tasks Block JavaScript

Long-running synchronous operations block the Event Loop. Async operations can't run until the call stack is empty:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Event Loop</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Long Tasks Block JavaScript</h4>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML += "1. Start<br>";

// This timeout should fire after 0ms
setTimeout(function() {
  document.getElementById("demo").innerHTML += "4. setTimeout (delayed by blocking code)<br>";
}, 0);

// Blocking operation - simulates 3 seconds of work
let start = Date.now();
while (Date.now() < start + 3000) {
  // Blocking loop
}

document.getElementById("demo").innerHTML += "2. After blocking loop (3 seconds later)<br>";

document.getElementById("demo").innerHTML += "3. End of synchronous code<br>";
</script>

</body>
</html>
```

Example 5

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript Event Loop