JavaScript Async

Back to JS page

Table of Contents

- JavaScript Async
  - Synchronous JavaScript
    - Example 1
  - Function Sequence
    - Example 2
  - Blocking Code
    - Example 3
  - Common Asynchronous Operations
    - Example 4
  - Document
  - Reference

Synchronous JavaScript

Synchronous code runs in sequence. Each operation waits for the previous one to complete before executing:

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

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

<h4>Synchronous JavaScript</h4>
<p id="demo"></p>

<script>
function myFunction() {
  return "Hello from synchronous function!";
}

let result = myFunction();
document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

Example 1

Result View Example

Function Sequence

JavaScript functions are executed in the sequence they are called, not in the sequence they are defined:

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

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

<h4>Function Sequence</h4>
<p id="demo"></p>

<script>
function displayResult(result) {
  document.getElementById("demo").innerHTML = "Result: " + result;
}

function calculate(a, b) {
  return a * b;
}

// Functions execute in sequence
let product = calculate(5, 3);
displayResult(product);
</script>

</body>
</html>
```

Example 2

Result View Example

Blocking Code

Long-running synchronous operations can block code execution. This is why asynchronous programming is important:

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

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

<h4>Blocking Code</h4>
<p id="demo1"></p>
<p id="demo2"></p>

<script>
document.getElementById("demo1").innerHTML = "Start of script";

// Simulate a blocking operation
function blockingOperation() {
  let start = Date.now();
  let end = start;
  while (end < start + 2000) {
    end = Date.now();
  }
  return "Blocking operation finished after 2 seconds";
}

let result = blockingOperation();
document.getElementById("demo2").innerHTML = result;
</script>

</body>
</html>
```

Example 3

Result View Example

Common Asynchronous Operations

Common async operations include `setTimeout()` and `fetch()`. These do not block the execution flow:

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

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

<h4>Common Asynchronous Operations</h4>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>

<script>
// This runs first
document.getElementById("demo1").innerHTML = "1. Start of script";

// setTimeout is asynchronous - it doesn't block
setTimeout(function() {
  document.getElementById("demo2").innerHTML = "3. Timeout finished (after 1 second)";
}, 1000);

// This runs second (not blocked by setTimeout)
document.getElementById("demo3").innerHTML = "2. End of script (not blocked)";
</script>

</body>
</html>
```

Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript Async