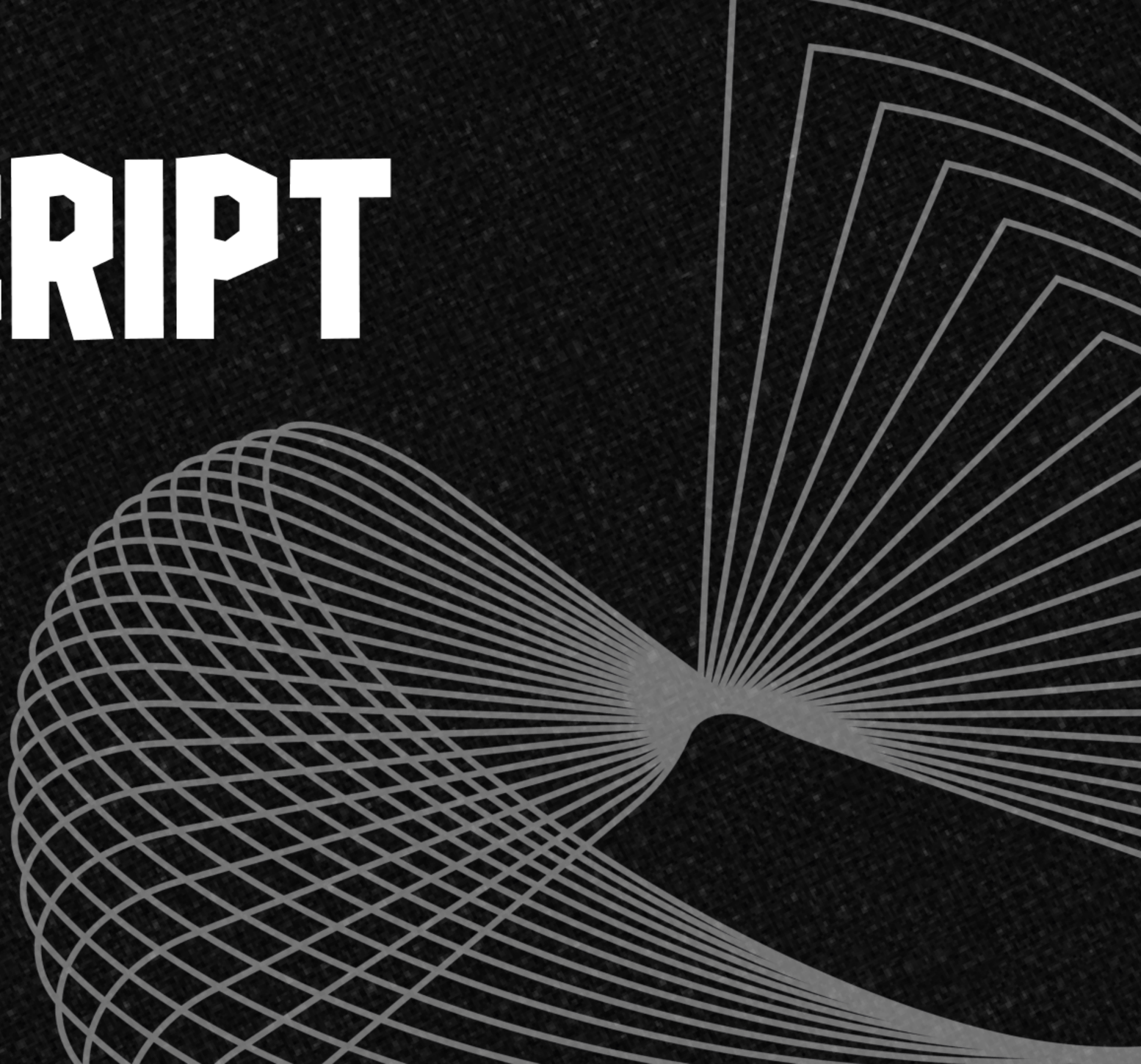
NodeJS

LABA

JAVASCRIPT EVENT LOOP



Javascript is single-threaded

Javascript is single-threaded

It can only execute a single task at a time.

Javascript is single-threaded

It can only execute a single task at a time.

The event loop is responsible for managing the execution of synchronous and asynchronous tasks.

Javascript is single-threaded

It can only execute a single task at a time.

The event loop is responsible for managing the execution of synchronous and asynchronous tasks.

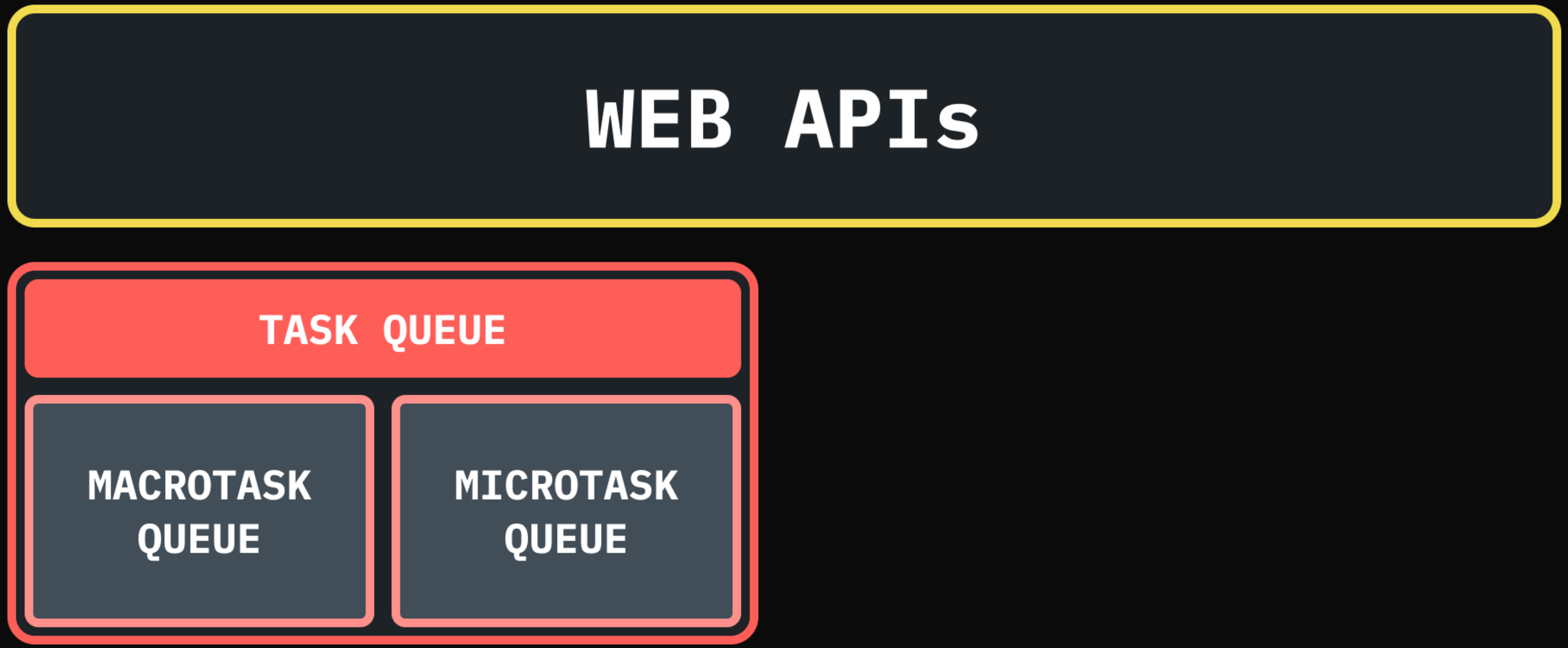
Allowing Javascript to handle non-blocking operations.

Event loop components

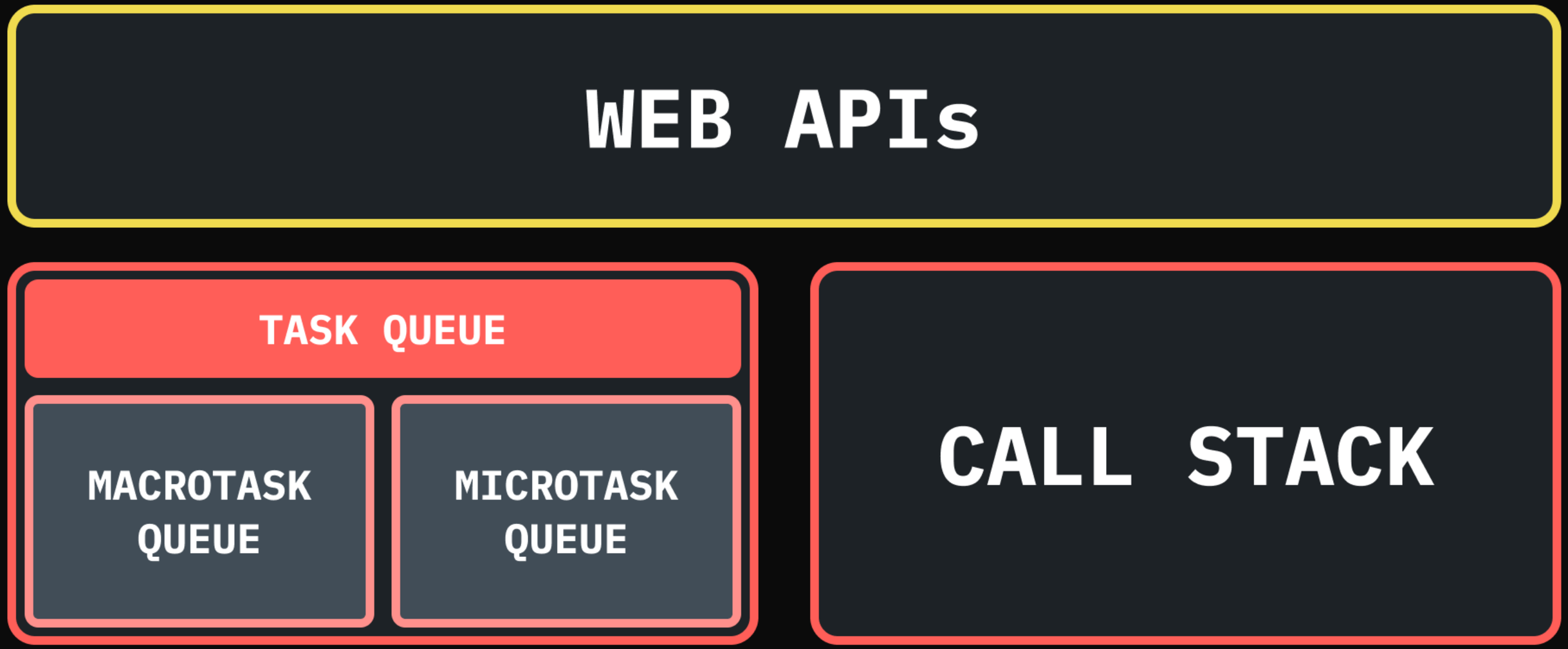
Event loop components

WEB APIs

Event loop components



Event loop components



Event Loop

JS

Javascript event loop visualization

```
function logSecond() {
    console.log("Second!");
}

function logThird() {
    console.log("Third!");
}

console.log("First!");

setTimeout(logSecond, 0);

Promise.resolve().then(logThird);

console.log("Fourth!");

```

call stack

task queue

microtask queue

console

Event Loop

JS

Javascript event loop visualization

```
function logSecond() {
    console.log("Second!");
}

function logThird() {
    console.log("Third!");
}

console.log("First!");

setTimeout(logSecond, 0);

Promise.resolve().then(logThird);

console.log("Fourth!");
```

task queue

```
console.log("First!");
```

call stack

microtask queue

console

Event Loop

Javascript event loop visualization

```
1  function logSecond() {
2      console.log("Second!");
3  }
4
5  function logThird() {
6      console.log("Third!");
7  }
8
9  console.log("First!");
10
11 setTimeout(logSecond, 0);
12
13 Promise.resolve().then(logThird);
14
15 console.log("Fourth!");
16
```

call stack

task queue

```
logSecond();
```

microtask queue

console

```
First!
```

Event Loop

Javascript event loop visualization

```
function logSecond() {
    console.log("Second!");
}

function logThird() {
    console.log("Third!");
}

console.log("First!");

setTimeout(logSecond, 0);

Promise.resolve().then(logThird);

console.log("Fourth!");
```

call stack

task queue

```
logSecond();
```

microtask queue

```
logThird();
```

console

```
First!
```

Event Loop

JS

Javascript event loop visualization

```
function logSecond() {
    console.log("Second!");
}

function logThird() {
    console.log("Third!");
}

console.log("First!");

setTimeout(logSecond, 0);

Promise.resolve().then(logThird);

console.log("Fourth!");
```

task queue

```
logSecond();
```

call stack

```
console.log("Fourth!");
```

microtask queue

```
logThird();
```

console

```
First!
```

Event Loop

JS

Javascript event loop visualization

```
1  function logSecond() {
2      console.log("Second!");
3  }
4
5  function logThird() {
6      console.log("Third!");
7  }
8
9  console.log("First!");
10
11 setTimeout(logSecond, 0);
12
13 Promise.resolve().then(logThird);
14
15 console.log("Fourth!");
16
```

task queue

```
logSecond();
```

call stack

```
console.log("Third!");
```

microtask queue

console

```
First!
Fourth!
```

Event Loop

JS

Javascript event loop visualization

```
function logSecond() {
    console.log("Second!");
}

function logThird() {
    console.log("Third!");
}

console.log("First!");

setTimeout(logSecond, 0);

Promise.resolve().then(logThird);

console.log("Fourth!");
```

task queue

```
console.log("Second!");
```

call stack

microtask queue

```
First!
Fourth!
Third!
```

console

Event Loop

JS

Javascript event loop visualization

```
function logSecond() {
    console.log("Second!");
}

function logThird() {
    console.log("Third!");
}

console.log("First!");

setTimeout(logSecond, 0);

Promise.resolve().then(logThird);

console.log("Fourth!");
```

task queue

call stack

microtask queue

```
First!
Fourth!
Third!
Second!
```

console