

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000000 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000001 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

Date.now

demo.js

Run Script

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000001 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000002 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000003 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000004 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000005 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000005 ms

Web/Node APIs

timeout

exp: 000005 ms
cb: handlerA

Stack

demo.js

Microtask Queue

Task Queue

Run Script

Console


```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000005 ms

Web/Node APIs

timeout

exp: 000005 ms
cb: handlerA

Stack

Microtask Queue

Task Queue

demo.js

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000005 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000006 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000007 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

demo.js

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000007 ms

Web/Node APIs

Stack

demo.js

Microtask Queue

settled: "val"
cb: handlerB

Task Queue

Run Script cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000008 ms

Web/Node APIs

Stack

demo.js

Microtask Queue

settled: "val"
cb: handlerB

Task Queue

Run Script cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000009 ms

Web/Node APIs

Stack

demo.js

Microtask Queue

settled: "val"
cb: handlerB

Task Queue

Run Script

cb: handlerA

Console

000009 ms

Web/Node APIs

Stack

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

Task Queue

demo.js

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```



```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000010 ms

Web/Node APIs

Stack

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

Task Queue

demo.js

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000010 ms

Web/Node APIs

Stack

demo.js

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

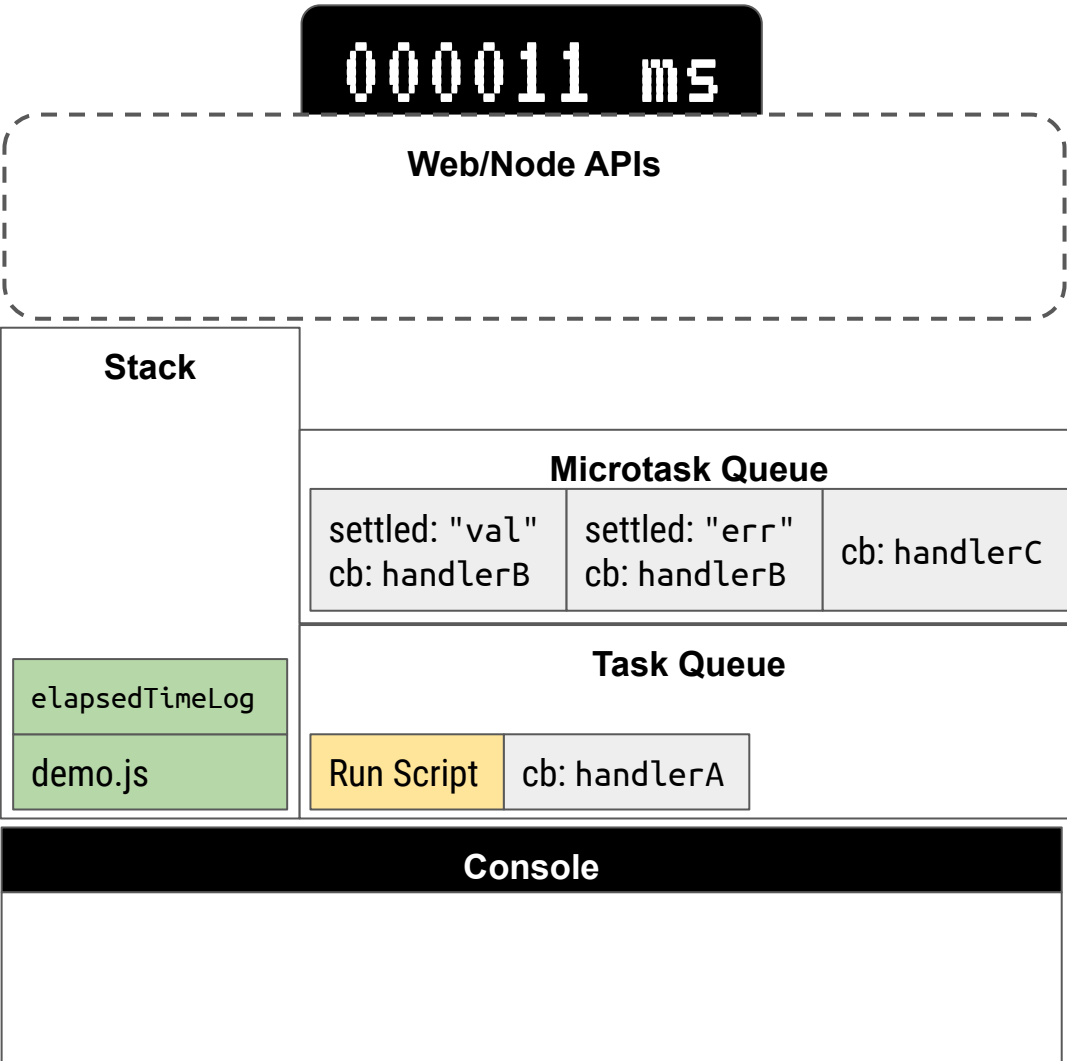
Task Queue

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```



000012 ms

Web/Node APIs

Stack

Date.now

elapsedTimeLog

demo.js

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000013 ms

Web/Node APIs

Stack

Math.floor

elapsedTimeLog

demo.js

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

Run Script

cb: handlerA

Console

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000014 ms

Web/Node APIs

Stack

console.log

elapsedTimeLog

demo.js

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

Run Script

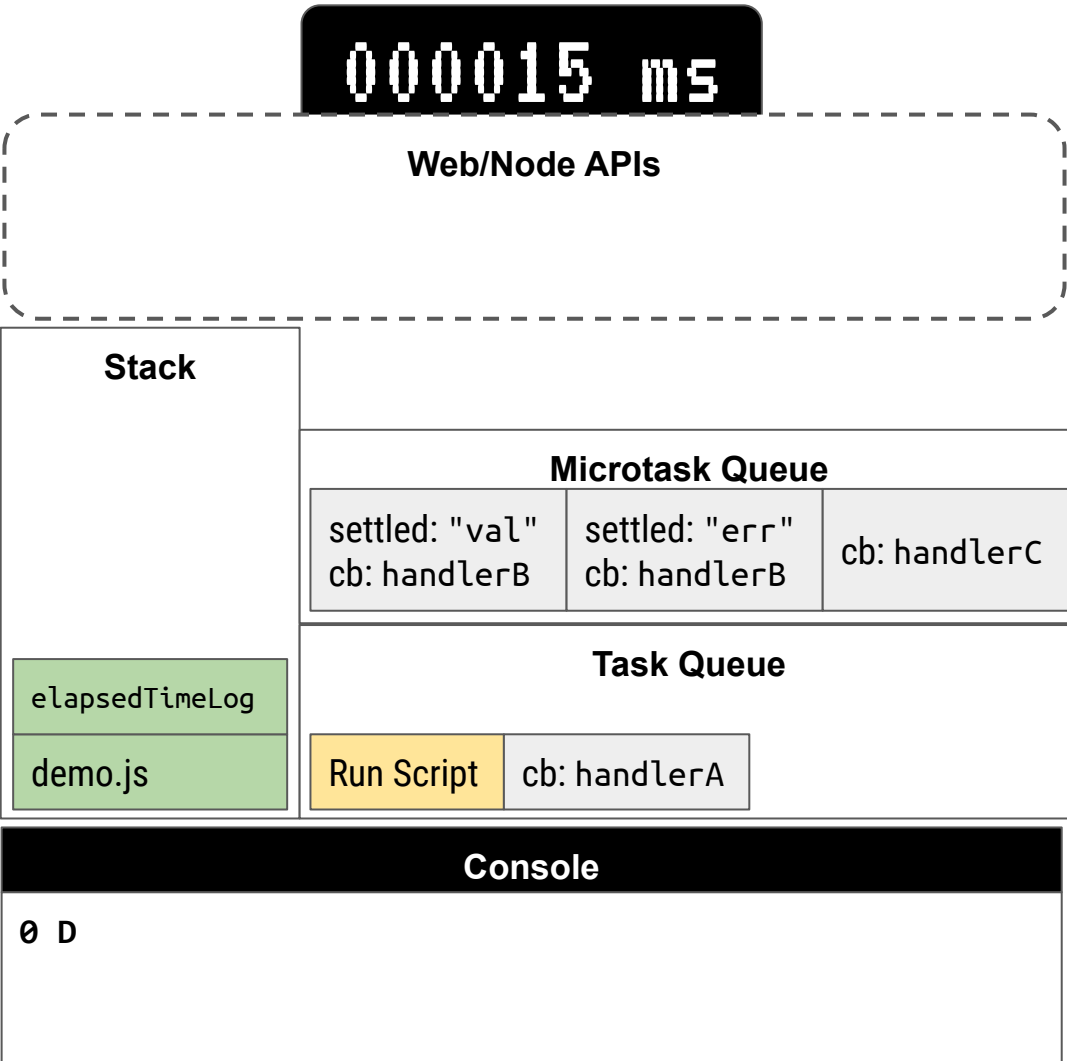
cb: handlerA

Console

0 D

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```



```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000016 ms

Web/Node APIs

Stack

Microtask Queue

settled: "val" cb: handlerB	settled: "err" cb: handlerB	cb: handlerC
--------------------------------	--------------------------------	--------------

Task Queue

demo.js	Run Script	cb: handlerA
---------	------------	--------------

Console

0 D

000017 ms

Web/Node APIs

Stack

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

Run Script

cb: handlerA

Console

0 D

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000018 ms

Web/Node APIs

Stack

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

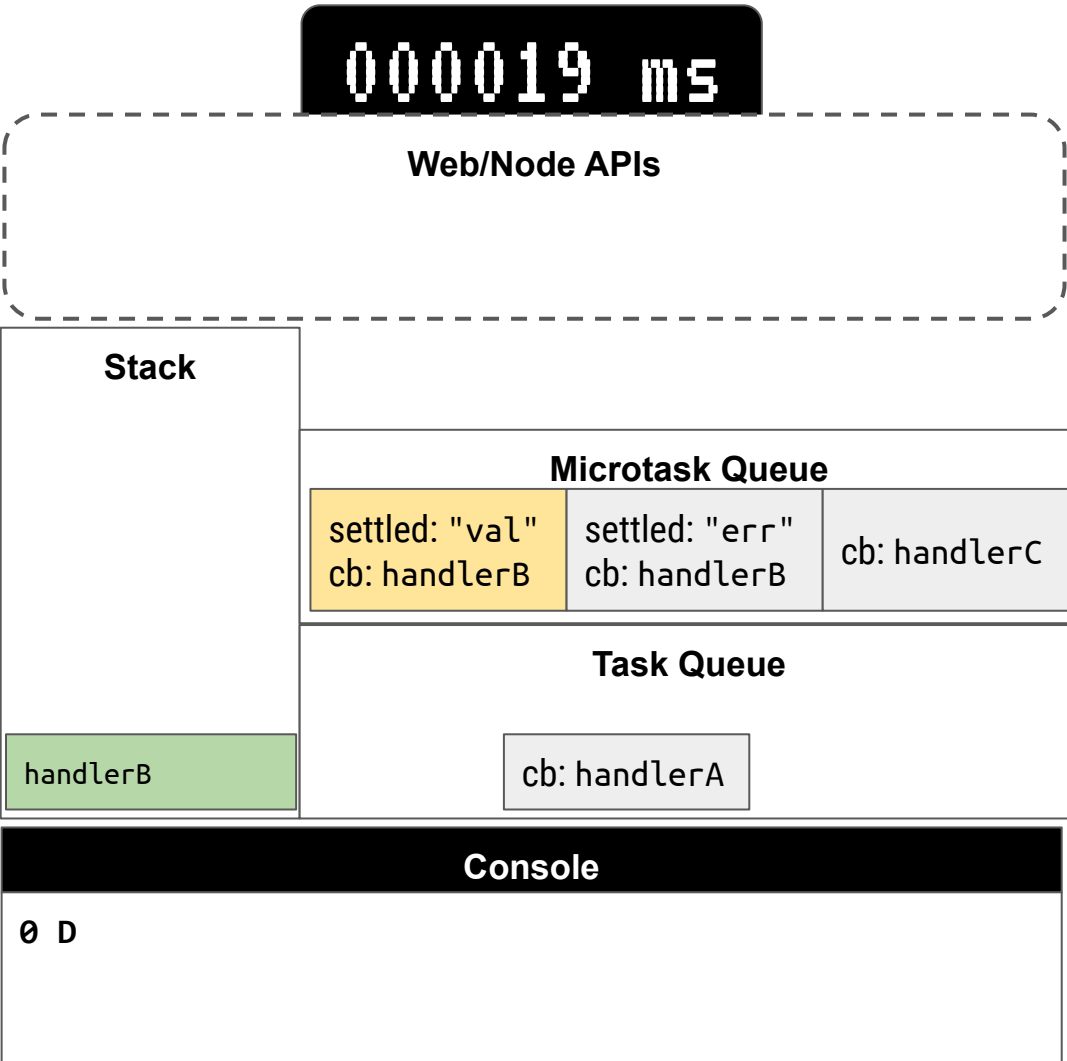
cb: handlerA

Console

0 D

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```



000023 ms

Web/Node APIs

Stack

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

console.log

elapsedTimeLog

handlerB

Task Queue

cb: handlerA

Console

0 D

0 val

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000026 ms

Web/Node APIs

Stack

Microtask Queue

settled: "val"
cb: handlerB

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

cb: handlerA

Console

0 D

0 val

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000027 ms

Web/Node APIs

Stack

Microtask Queue

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

cb: handlerA

Console

0 D

0 val

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000028 ms

Web/Node APIs

Stack

handlerB

Microtask Queue

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

cb: handlerA

Console

0 D

0 val

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000028 ms

Web/Node APIs

Stack

handlerB

Microtask Queue

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

cb: handlerA

Console

0 D

0 val

000032 ms

Web/Node APIs

Stack

Microtask Queue

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000035 ms

Web/Node APIs

Stack

Microtask Queue

settled: "err"
cb: handlerB

cb: handlerC

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000036 ms

Web/Node APIs

Stack

Microtask Queue

cb: handlerC

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

000037 ms

Web/Node APIs

Stack

handlerC

Microtask Queue

cb: handlerC

Task Queue

cb: handlerA

Console

0 D

0 val

0 err

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000037 ms

Web/Node APIs

Stack

handlerC

Microtask Queue

cb: handlerC

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

000041 ms

Web/Node APIs

Stack

Microtask Queue

cb: handlerC

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

0 C

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000044 ms

Web/Node APIs

Stack

Microtask Queue

cb: handlerC

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

0 C

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000045 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

0 C


```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000046 ms

Web/Node APIs

Stack

handlerA

Microtask Queue

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

0 C

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000046 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

handlerA

cb: handlerA

Console

0 D
0 val
0 err

0 C

000050 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

console.log

elapsedTimeLog

handlerA

cb: handlerA

Console

0 D

0 val

0 err

0 C

0 A

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```

000053 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

cb: handlerA

Console

0 D
0 val
0 err

0 C
0 A

000054 ms

Web/Node APIs

Stack

Microtask Queue

Task Queue

Console

0 D
0 val
0 err

0 C
0 A

```
01: const start = Date.now();
02: function elapsedTimeLog(...args) {
03:   const elapsedMs = Date.now() - start;
04:   const sec = Math.floor(elapsedMs / 1000);
05:   console.log(sec, ...args);
06: }
07:
08: function handlerA() {
09:   elapsedTimeLog("A");
10: }
11:
12: function handlerB(...args) {
13:   elapsedTimeLog(...args);
14: }
15:
16: function handlerC() {
17:   elapsedTimeLog("C");
18: }
19:
20: setTimeout(handlerA, 0);
21: Promise.resolve("val").then(handlerB);
22: Promise.reject("err").catch(handlerB);
23: queueMicrotask(handlerC);
24: elapsedTimeLog("D");
```