warmup.js

```
01: function waitSync(delayMs) {
02:   const waitStart = Date.now();
03:   while (Date.now() - waitStart < delayMs) {}
04: }
05:
06: const start = Date.now();
07: function elapsedTimeLog(...args) {
08:   const elapsedMs = Date.now() - start;
09:   const sec = Math.floor(elapsedMs / 1000);
10:   console.log(sec, ...args);
11: }
12:
13: function handlerA() {
14:   waitSync(6000);
15:   elapsedTimeLog("A");
16: }
17:
18: function handlerB() {
19:   elapsedTimeLog("B");
20: }
21:
22: setTimeout(handlerA, 1000);
23: setTimeout(handlerB, 2000);
24:
25: elapsedTimeLog("C");
```

demo.js

```
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");
```

exercise1.js

```
01: setTimeout(() => {
02:   Promise.resolve()
03:     .then(() => console.log("A"));
04:   console.log("B");
05: }, 0);
06:
07: Promise.resolve()
08:   .then(() => console.log("C"));
09:
10: console.log("D");
```

exercise2.js

```
01: const pr0 = Promise.resolve();
02: const pr1 = pr0.then(() => console.log("A"));
03: const pr2 = pr1.then(() => console.log("B"));
04:
05: const pr3 = Promise.resolve();
06: const pr4 = pr3.then(() => console.log("C"));
07: const pr5 = pr4.then(() => console.log("D"));
08:
09: console.log("E");
```

exercise3.js

```
01: Promise.resolve()
02:   .then(() => console.log("A"))
03:   .then(() => console.log("B"))
04:   .then(() => console.log("C"));
05:
06: Promise.reject()
07:   .then(() => console.log("D"))
08:   .catch(() => console.log("E"));
```

exercise4.js

```
01: setTimeout(() => console.log("A"), 0);
02:
03: queueMicrotask(() => {
04:   queueMicrotask(() => {
05:     queueMicrotask(() => {
06:       console.log("B");
07:     });
08:     console.log("C");
09:   });
10:   console.log("D");
11: });
12:
13: console.log("E");
```

interactive-logger.html

```
01: <!DOCTYPE html>
02: <html>
03:   <body>
04:     <textarea id="userText"></textarea>
05:     <button onclick="startLogging()">Start Log</button>
06:     <button onclick="stopLogging()">Stop Log</button>
07:   </body>
08:   <script>
09:     let toid = null;
10:
11:     function startLogging() {
12:       const v = document.getElementById("userText").value;
13:       console.log(v);
14:       toid = setTimeout(startLogging, 0);
15:     }
16:
17:     function stopLogging() {
18:       if (toid !== null) {
19:         clearTimeout(toid);
20:         toid = null;
21:       }
22:     }
23:   </script>
24: </html>
```

cache-client.js

```
01: import Cache from "./cache-v1.mjs";
02:
03: const cache = new Cache("memory");
04: cache.on(
05:   "connected",
06:   () => console.log("cache connected")
07: );
08: cache.on(
09:   "error",
10:   (err) => console.log(err)
11: );
```

cache-v1.mjs

```
01: import { EventEmitter } from "node:events";
02: import { open } from "fs/promises";
03:
04: export default class Cache extends EventEmitter {
05:   constructor(type) {
06:     super();
07:     switch (type) {
08:       case "memory":
09:         this._memCache = new Map();
10:         this.emit("connected");
11:         break;
12:       case "file":
13:         open("./cache-file.txt", "w+")
14:           .then((fileHandle) => {
15:             this._fileCache = fileHandle;
16:             this.emit("connected");
17:           })
18:           .catch((err) => this.emit("error", err));
19:         break;
20:       default:
21:         throw new Error("invalid type");
22:     }
23:   }
24: }
```