/\*\*

\* The `timers/promises` API provides an alternative set of timer functions

\* that return `Promise` objects. The API is accessible via`require('timers/promises')`.

\*

\* ```js

\* import {

\* setTimeout,

\* setImmediate,

\* setInterval,

\* } from 'timers/promises';

\* ```

\* @since v15.0.0

\*/

declare module 'timers/promises' {

import { TimerOptions } from 'node:timers';

/\*\*

\* ```js

\* import {

\* setTimeout,

\* } from 'timers/promises';

\*

\* const res = await setTimeout(100, 'result');

\*

\* console.log(res); // Prints 'result'

\* ```

\* @since v15.0.0

\* @param [delay=1] The number of milliseconds to wait before fulfilling the promise.

\* @param value A value with which the promise is fulfilled.

\*/

function setTimeout<T = void>(delay?: number, value?: T, options?: TimerOptions): Promise<T>;

/\*\*

\* ```js

\* import {

\* setImmediate,

\* } from 'timers/promises';

\*

\* const res = await setImmediate('result');

\*

\* console.log(res); // Prints 'result'

\* ```

\* @since v15.0.0

\* @param value A value with which the promise is fulfilled.

\*/

function setImmediate<T = void>(value?: T, options?: TimerOptions): Promise<T>;

/\*\*

\* Returns an async iterator that generates values in an interval of `delay` ms.

\*

\* ```js

\* import {

\* setInterval,

\* } from 'timers/promises';

\*

\* const interval = 100;

\* for await (const startTime of setInterval(interval, Date.now())) {

\* const now = Date.now();

\* console.log(now);

\* if ((now - startTime) > 1000)

\* break;

\* }

\* console.log(Date.now());

\* ```

\* @since v15.9.0

\*/

function setInterval<T = void>(delay?: number, value?: T, options?: TimerOptions): AsyncIterable<T>;

}

declare module 'node:timers/promises' {

export \* from 'timers/promises';

}