/\*\*

\* @since v0.3.7

\*/

declare module 'module' {

import { URL } from 'node:url';

namespace Module {

/\*\*

\* The `module.syncBuiltinESMExports()` method updates all the live bindings for

\* builtin `ES Modules` to match the properties of the `CommonJS` exports. It

\* does not add or remove exported names from the `ES Modules`.

\*

\* ```js

\* const fs = require('fs');

\* const assert = require('assert');

\* const { syncBuiltinESMExports } = require('module');

\*

\* fs.readFile = newAPI;

\*

\* delete fs.readFileSync;

\*

\* function newAPI() {

\* // ...

\* }

\*

\* fs.newAPI = newAPI;

\*

\* syncBuiltinESMExports();

\*

\* import('fs').then((esmFS) => {

\* // It syncs the existing readFile property with the new value

\* assert.strictEqual(esmFS.readFile, newAPI);

\* // readFileSync has been deleted from the required fs

\* assert.strictEqual('readFileSync' in fs, false);

\* // syncBuiltinESMExports() does not remove readFileSync from esmFS

\* assert.strictEqual('readFileSync' in esmFS, true);

\* // syncBuiltinESMExports() does not add names

\* assert.strictEqual(esmFS.newAPI, undefined);

\* });

\* ```

\* @since v12.12.0

\*/

function syncBuiltinESMExports(): void;

/\*\*

\* `path` is the resolved path for the file for which a corresponding source map

\* should be fetched.

\* @since v13.7.0, v12.17.0

\*/

function findSourceMap(path: string, error?: Error): SourceMap;

interface SourceMapPayload {

file: string;

version: number;

sources: string[];

sourcesContent: string[];

names: string[];

mappings: string;

sourceRoot: string;

}

interface SourceMapping {

generatedLine: number;

generatedColumn: number;

originalSource: string;

originalLine: number;

originalColumn: number;

}

/\*\*

\* @since v13.7.0, v12.17.0

\*/

class SourceMap {

/\*\*

\* Getter for the payload used to construct the `SourceMap` instance.

\*/

readonly payload: SourceMapPayload;

constructor(payload: SourceMapPayload);

/\*\*

\* Given a line number and column number in the generated source file, returns

\* an object representing the position in the original file. The object returned

\* consists of the following keys:

\*/

findEntry(line: number, column: number): SourceMapping;

}

}

interface Module extends NodeModule {}

class Module {

static runMain(): void;

static wrap(code: string): string;

static createRequire(path: string | URL): NodeRequire;

static builtinModules: string[];

static Module: typeof Module;

constructor(id: string, parent?: Module);

}

global {

interface ImportMeta {

url: string;

/\*\*

\* @experimental

\* This feature is only available with the `--experimental-import-meta-resolve`

\* command flag enabled.

\*

\* Provides a module-relative resolution function scoped to each module, returning

\* the URL string.

\*

\* @param specified The module specifier to resolve relative to `parent`.

\* @param parent The absolute parent module URL to resolve from. If none

\* is specified, the value of `import.meta.url` is used as the default.

\*/

resolve?(specified: string, parent?: string | URL): Promise<string>;

}

}

export = Module;

}

declare module 'node:module' {

import module = require('module');

export = module;

}