Let's use `await` at the top level in a module `db-connection.js`. This makes sense since the connection to the DB needs to be established before the module is usable.

db-connection.js

```
const connectToDB = async url => {
    await new Promise(r => setTimeout(r, 1000));
};

// This is a top-level-await
await connectToDB("my-sql://example.com");

export const dbCall = async data => {
    // This is a normal await, because it's in an async function
    await new Promise(r => setTimeout(r, 100));
    return "fake data";
};

export const close = () => {
    console.log("closes the DB connection");
};
```

But `db-connection.js` is no longer a normal module now. It's an **async module** now. Async modules have a different evaluation semantics. While normal modules evaluate synchronously, async modules evaluate asynchronously.

Async modules can still be imported with a normal `import`. But importing an async module makes the importing module also an async module.

The `import`s still hoist and are evaluated in parallel.

Tree shaking still works as usual. Here the `close` function is never used and will be removed from the output bundle in production mode.

UserApi.js

```
import { dbCall } from "./db-connection.js";

export const createUser = async name => {
    command = `CREATE USER ${name}`;
    // This is a normal await, because it's in an async function
    await dbCall({ command });
};
```

Now it looks like that this pattern will continue and will infect all using modules as async modules.

Yes, this is kind of true and makes sense. All these modules have their evaluation semantics changed to be async.

But you as a developer don't want this. You want to break the chain at a point in your module graph where it makes sense. Luckily there is a nice way to break the chain.

You can use `import("./UserApi.js")` to import the module instead of `import`. As this returns a Promise it can be awaited to wait for module evaluation (including top-level-awaits) and handle failures.

Handling failures is an important point here. When using top-level-await there are more ways that a module evaluation can fail now. In this example connecting to the DB may fail.

Actions.js

```
// import() doesn't care about whether a module is an async module or not
const UserApi = import("./UserApi.js");

export const CreateUserAction = async name => {
    // These are normal awaits, because they are in an async function
    const { createUser } = await UserApi;
    await createUser(name);
};

// You can place import() where you like
// Placing it at top-level will start loading and evaluating on
//   module evaluation.
//   see CreateUserAction above
//   Here: Connecting to the DB starts when the application starts
// Placing it inside of an (async) function will start loading
//   and evaluating when the function is called for the first time
//   which basically makes it lazy-loaded.
//   see AlternativeCreateUserAction below
//   Here: Connecting to the DB starts when AlternativeCreateUserAction
//         is called
export const AlternativeCreateUserAction = async name => {
    const { createUser } = await import("./UserApi.js");
    await createUser(name);
};

// Note: Using await import() at top-level doesn't make much sense
//       except in rare cases. It will import modules sequentially.
```

As `Actions.js` doesn't use any top-level-await nor `import`s an async module directly so it's not an async module.

example.js

```
import { CreateUserAction } from "./Actions.js";

(async ()=> {
```

```
    await CreateUserAction("John");
})();
```

As a guideline, you should prevent your application entry point to become an async module when compiling for web targets. Doing async actions at application bootstrap will delay your application startup and may be negative for UX. Use `import()` to do async action on-demand or in the background and use spinners or other indicators to inform the user about background actions.

When compiling for other targets like node.js, electron or WebWorkers, it may be fine that your entry point becomes an async module.

dist/output.js

```
/******/ (() => { // webpackBootstrap
/******/ 	"use strict";
/******/ 	var __webpack_modules__ = ([
/* 0 */,
/* 1 */
/*!*********************!*\
  !*** ./Actions.js ***!
  \*********************/
/*! namespace exports */
/*! export AlternativeCreateUserAction [provided] [no usage info] [missing usage info prevents renaming] */
/*! export CreateUserAction [provided] [no usage info] [missing usage info prevents renaming] */
/*! other exports [not provided] [no usage info] */
/*! runtime requirements: __webpack_require__.r, __webpack_exports__, __webpack_require__.e, __webpack_require__, __webpack_require__.d, __webpack_require__.* */
/***/ ((__unused_webpack_module, __webpack_exports__, __webpack_require__) => {

__webpack_require__.r(__webpack_exports__);
/* harmony export */ __webpack_require__.d(__webpack_exports__, {
/* harmony export */   "CreateUserAction": () => (/* binding */ CreateUserAction),
/* harmony export */   "AlternativeCreateUserAction": () => (/* binding */ AlternativeCreateUserAction)
/* harmony export */ });
// import() doesn't care about whether a module is an async module or not
const UserApi = __webpack_require__.e(/*! import() */ 497).then(__webpack_require__.bind(__webpack_require__, /*! ./UserApi.js */ 2));

const CreateUserAction = async name => {
    // These are normal awaits, because they are in an async function
    const { createUser } = await UserApi;
    await createUser(name);
};

// You can place import() where you like
// Placing it at top-level will start loading and evaluating on
```

```
//   module evaluation.
//   see CreateUserAction above
//   Here: Connecting to the DB starts when the application starts
// Placing it inside of an (async) function will start loading
//   and evaluating when the function is called for the first time
//   which basically makes it lazy-loaded.
//   see AlternativeCreateUserAction below
//   Here: Connecting to the DB starts when AlternativeCreateUserAction
//         is called
const AlternativeCreateUserAction = async name => {
    const { createUser } = await __webpack_require__.e(/*! import() */
497).then(__webpack_require__.bind(__webpack_require__, /*! ./UserApi.js */ 2));
    await createUser(name);
};

// Note: Using await import() at top-level doesn't make much sense
//       except in rare cases. It will import modules sequentially.


/***/ })
/******/        ]);
```

▶ /* webpack runtime code */

```
var __webpack_exports__ = {};
// This entry need to be wrapped in an IIFE because it need to be isolated against
other modules in the chunk.
(() => {
/*!*********************!*\
  !*** ./example.js ***!
  \*********************/
/*! namespace exports */
/*! exports [not provided] [no usage info] */
/*! runtime requirements: __webpack_require__, __webpack_require__.r,
__webpack_exports__, __webpack_require__.* */
__webpack_require__.r(__webpack_exports__);
/* harmony import */ var _Actions_js__WEBPACK_IMPORTED_MODULE_0__ =
__webpack_require__(/*! ./Actions.js */ 1);


(async ()=> {
    await (0,_Actions_js__WEBPACK_IMPORTED_MODULE_0__.CreateUserAction)("John");
})();

})();

/******/ })()
;
```

dist/497.output.js

```
"use strict";
(self["webpackChunk"] = self["webpackChunk"] || []).push([[497],[
/* 0 */,
/* 1 */,
/* 2 */
/*!*********************!*\
  !*** ./UserApi.js ***!
  \*********************/
/*! namespace exports */
/*! export createUser [provided] [no usage info] [missing usage info prevents
renaming] */
/*! other exports [not provided] [no usage info] */
/*! runtime requirements: __webpack_require__, __webpack_require__.r,
__webpack_exports__, module, __webpack_require__.a, __webpack_require__.d,
__webpack_require__.* */
/***/ ((module, __webpack_exports__, __webpack_require__) => {

__webpack_require__.a(module, async (__webpack_handle_async_dependencies__) => {
__webpack_require__.r(__webpack_exports__);
/* harmony export */ __webpack_require__.d(__webpack_exports__, {
/* harmony export */   "createUser": () => (/* binding */ createUser)
/* harmony export */ });
/* harmony import */ var _db_connection_js__WEBPACK_IMPORTED_MODULE_0__ =
__webpack_require__(/*! ./db-connection.js */ 3);
var __webpack_async_dependencies__ =
__webpack_handle_async_dependencies__([_db_connection_js__WEBPACK_IMPORTED_MODULE_0__]);

_db_connection_js__WEBPACK_IMPORTED_MODULE_0__ =
(__webpack_async_dependencies__.then ? await __webpack_async_dependencies__ :
__webpack_async_dependencies__)[0];


const createUser = async name => {
    command = `CREATE USER ${name}`;
    // This is a normal await, because it's in an async function
    await (0,_db_connection_js__WEBPACK_IMPORTED_MODULE_0__.dbCall)({ command });
};

});

/***/ }),
/* 3 */
/*!**************************!*\
  !*** ./db-connection.js ***!
  \**************************/
/*! namespace exports */
/*! export close [provided] [no usage info] [missing usage info prevents renaming]
*/
```

```
/*! export dbCall [provided] [no usage info] [missing usage info prevents renaming]
*/
/*! other exports [not provided] [no usage info] */
/*! runtime requirements: __webpack_require__.r, __webpack_exports__, module,
__webpack_require__.a, __webpack_require__.d, __webpack_require__.* */
/***/ ((module, __webpack_exports__, __webpack_require__) => {

__webpack_require__.a(module, async (__webpack_handle_async_dependencies__) => {
__webpack_require__.r(__webpack_exports__);
/* harmony export */ __webpack_require__.d(__webpack_exports__, {
/* harmony export */   "dbCall": () => (/* binding */ dbCall),
/* harmony export */   "close": () => (/* binding */ close)
/* harmony export */ });
const connectToDB = async url => {
    await new Promise(r => setTimeout(r, 1000));
};

// This is a top-level-await
await connectToDB("my-sql://example.com");

const dbCall = async data => {
    // This is a normal await, because it's in an async function
    await new Promise(r => setTimeout(r, 100));
    return "fake data";
};

const close = () => {
    console.log("closes the DB connection");
};

__webpack_handle_async_dependencies__();
}, 1);

/***/ })
]]);
```

in production mode:

```
"use strict";(self.webpackChunk=self.webpackChunk||[]).push([[497],{497:(a,e,s)=>
{s.a(a,(async a=>{s.r(e),s.d(e,{createUser:()=>c});var t=s(447),n=a([t]);t=(n.then?
await n:n)[0];const c=async a=>{command=`CREATE USER ${a}`,await(0,t.j)
({command})}}))},447:(a,e,s)=>{s.a(a,(async a=>{s.d(e,{j:()=>t}),await(async a=>
{await new Promise((a=>setTimeout(a,1e3)))})();const t=async a=>(await new
Promise((a=>setTimeout(a,100))),"fake data");a()}),1)}}]);
```

Info

Unoptimized

```
asset output.js 15.2 KiB [emitted] (name: main)
asset 497.output.js 2.8 KiB [emitted]
chunk (runtime: main) output.js (main) 1.19 KiB (javascript) 7.7 KiB (runtime) [entry]
[rendered]
  > ./example.js main
  runtime modules 7.7 KiB 9 modules
  dependent modules 1.09 KiB [dependent] 1 module
  ./example.js 103 bytes [built] [code generated]
    [no exports]
    [used exports unknown]
    entry ./example.js main
chunk (runtime: main) 497.output.js 617 bytes [rendered]
  > ./UserApi.js ./Actions.js 22:30-52
  > ./UserApi.js ./Actions.js 2:16-38
  dependent modules 402 bytes [dependent] 1 module
  ./UserApi.js 215 bytes [built] [code generated]
    [exports: createUser]
    [used exports unknown]
    import() ./UserApi.js ./Actions.js 2:16-38
    import() ./UserApi.js ./Actions.js 22:30-52
webpack 5.51.1 compiled successfully
```

Production mode

```
asset output.js 2.88 KiB [emitted] [minimized] (name: main)
asset 497.output.js 448 bytes [emitted] [minimized]
chunk (runtime: main) output.js (main) 1.19 KiB (javascript) 7.7 KiB (runtime) [entry]
[rendered]
  > ./example.js main
  runtime modules 7.7 KiB 9 modules
  ./example.js + 1 modules 1.19 KiB [built] [code generated]
    [no exports]
    [no exports used]
    entry ./example.js main
chunk (runtime: main) 497.output.js 617 bytes [rendered]
  > ./UserApi.js ./Actions.js 22:30-52
  > ./UserApi.js ./Actions.js 2:16-38
  dependent modules 402 bytes [dependent] 1 module
  ./UserApi.js 215 bytes [built] [code generated]
    [exports: createUser]
    import() ./UserApi.js ./example.js + 1 modules ./Actions.js 2:16-38
    import() ./UserApi.js ./example.js + 1 modules ./Actions.js 22:30-52
webpack 5.51.1 compiled successfully
```