This example shows how the `sideEffects` flag for library authors works.

The example contains a large library, `big-module`. `big-module` contains multiple child modules: `a`, `b` and `c`. The exports from the child modules are re-exported in the entry module (`index.js`) of the library. A consumer uses **some** of the exports, importing them from the library via `import { a, b } from "big-module"`. According to the EcmaScript spec, all child modules *must* be evaluated because they could contain side effects.

The `"sideEffects": false` flag in `big-module`'s `package.json` indicates that the package's modules have no side effects (on evaluation) and only expose exports. This allows tools like webpack to optimize re-exports. In the case `import { a, b } from "big-module-with-flag"` is rewritten to `import { a } from "big-module-with-flag/a"; import { b } from "big-module-with-flag/b"`.

The example contains two variants of `big-module`. `big-module` has no `sideEffects` flag and `big-module-with-flag` has the `sideEffects` flag. The example client imports `a` and `b` from each of the variants.

After being built by webpack, the output bundle contains `index.js` `a.js` `b.js` `c.js` from `big-module`, but only `a.js` and `b.js` from `big-module-with-flag`.

Advantages:

- Smaller bundles
- Faster boot up

example.js

{{example.js}}

node_modules/big-module/package.json

_{{node_modules/big-module/package.json}}_

node_modules/big-module-with-flag/package.json

_{{node_modules/big-module-with-flag/package.json}}_

node_modules/big-module(-with-flag)/index.js

_{{node_modules/big-module-with-flag/index.js}}_

dist/output.js

{{dist/output.js}}

Info

Unoptimized

{{stdout}}

Production mode

{{production:stdout}}