This example demonstrates how to build a complex library with webpack. The library consists of multiple parts that are usable on its own and together.

When using this library with script tags it exports itself to the namespace `MyLibrary` and each part to a property in this namespace (`MyLibrary.alpha` and `MyLibrary.beta`). When consuming the library with CommonsJS or AMD it just exports each part.

We are using multiple entry points (`entry` option) to build every part of the library as a separate output file. The `output.filename` option contains `[name]` to give each output file a different name.

We are using the `libraryTarget` option to generate a UMD (Universal Module Definition) module that is consumable in CommonsJS, AMD and with script tags. The `library` option defines the namespace. We are using `[name]` in the `library` option to give every entry a different namespace.

You can see that webpack automatically wraps your module so that it is consumable in every environment. All you need is this simple config.

Note: You can also use the `library` and `libraryTarget` options without multiple entry points. Then you don't need `[name]` .

Note: When your library has dependencies that should not be included in the compiled version, you can use the `externals` option. See externals example.

webpack.config.js

```
_{{webpack.config.js}}_
```

dist/MyLibrary.alpha.js

```
_{{dist/MyLibrary.alpha.js}}_
```

dist/MyLibrary.beta.js

```
_{{dist/MyLibrary.beta.js}}_
```

Info

Unoptimized

```
_{{stdout}}_
```

Production mode

```
_{{production:stdout}}_
```