

This example shows how to create an explicit vendor chunk as well as a common chunk for code shared among entry points. In this example, we have 3 entry points: `pageA` , `pageB` , and `pageC` . Those entry points share some of the same utility modules, but not others. This configuration will pull out any modules common to at least 2 bundles and place it in the `common` bundle instead, all while keeping the specified vendor libraries in their own bundle by themselves.

To better understand, here are the entry points and which utility modules they depend on:

- `pageA`
 - `utility1`
 - `utility2`
- `pageB`
 - `utility2`
 - `utility3`
- `pageC`
 - `utility2`
 - `utility3`

Given this configuration, webpack will produce the following bundles:

- `vendor`
 - `webpack runtime`
 - `vendor1`
 - `vendor2`
- `common`
 - `utility2`
 - `utility3`
- `pageA`
 - `pageA`
 - `utility1`
- `pageB`
 - `pageB`
- `pageC`
 - `pageC`

With this bundle configuration, you would load your third party libraries, then your common application code, then your page-specific application code.

webpack.config.js

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

dist/vendor.js

```
_{{dist/vendor.js}}_
```

dist/commons-utility2_js.js

```
_{{dist/commons-utility2_js.js}}_
```

dist/commons-utility3_js.js

```
_{{dist/commons-utility3_js.js}}_
```

dist/pageA.js

```
_{{dist/pageA.js}}_
```

dist/pageB.js

```
_{{dist/pageB.js}}_
```

dist/pageC.js

```
_{{dist/pageC.js}}_
```

Info

Unoptimized

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

Production mode

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