Info

This example illustrates webpack's algorithm for automatic deduplication using `optimization.splitChunks`.

This example application contains 7 pages, each importing 1-3 modules from the `node_modules` folder (vendor libs) and 0-3 modules from the `stuff` folder (application modules). In reality, an application is probably more complex, but the same mechanisms apply.

The following configuration is used:

- `optimization.splitChunks.chunks: "all"` - This opt-in into automatic splitting of initial chunks which is off by default
- `optimization.splitChunks.maxInitial/AsyncRequests: 20` - This opt-in into an HTTP2 optimized splitting mode by increasing the allowed amount of requests. The browser only supports 6 requests in parallel for HTTP1.1.

Interpreting the result

- `pageA.js` the normal output files for the entrypoint `pageA`
- `vendors~pageD~pageE~pageF~pageG.js` vendor libs shared by these pages extracted into a separate output file when larger than the threshold in size
- `vendors~pageA.js` vendors only used by a single page but larger than the threshold in size
- `pageA~pageD~pageF.js` application modules shared by these pages and larger than the threshold in size

Here, the threshold is 40 bytes but by default (in a real application) 30kb.

Some modules are intentionally duplicated, i. e. `./stuff/s4.js` is shared by `pageA` and `pageC`, but it's the only shared module so no separate output file is created because it would be smaller than the threshold. A separate request (which comes with an overhead and worsen gzipping) is not worth the extra bytes.

Note: decreasing `maxInitial/AsyncRequest` will increase duplication further to reduce the number of requests. Duplication doesn't affect the initial page load, it only affects download size of navigations to other pages of the application.

webpack.config.js

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

Production mode

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