This example shows how to use multiple entry points with a commons chunk.

In this example, you have two (HTML) pages `pageA` and `pageB`. You want to create individual bundles for each page. In addition to this, you want to create a shared bundle that contains all the modules used in both pages (assuming there are many/big modules in common). The pages also use Code Splitting to load a less used part of the features on demand.

You can see how to define multiple entry points via the `entry` option.

You can use

You can see the output files:

- `commons.js` contains:
  - module `common.js` which is used in both pages
- `pageA.js` contains: (`pageB.js` is similar)
  - the module system
  - chunk loading logic
  - the entry point `pageA.js`
  - it would contain any other module that is only used by `pageA`
- `406.js` is an additional chunk which is used by both pages. It contains:
  - module `shared.js`

You can also see the info that is printed to console. It shows among others:

- the generated files
- the chunks with file, name, and id
  - see lines starting with `chunk`
- the modules that are in the chunks
- the reasons why the modules are included
- the reasons why a chunk is created
  - see lines starting with `>`

pageA.js

```
_{{pageA.js}}_
```

pageB.js

```
_{{pageB.js}}_
```

webpack.config.js

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

pageA.html

```
_{{pageA.html}}_
```

dist/commons.js

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

dist/pageA.js

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

dist/pageB.js

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

dist/52.js

```
_{{dist/52.js}}_
```

Info

Unoptimized

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

Production mode

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