A common challenge with combining `[chunkhash]` and Code Splitting is that the entry chunk includes the webpack runtime and with it the chunkhash mappings. This means it's always updated and the `[chunkhash]` is pretty useless because this chunk won't be cached.

A very simple solution to this problem is to create another chunk that contains only the webpack runtime (including chunkhash map). This can be achieved with `optimization.runtimeChunk` options. To avoid the additional request for another chunk, this pretty small chunk can be inlined into the HTML page.

The configuration required for this is:

- use `[chunkhash]` in `output.filename` (Note that this example doesn't do this because of the example generator infrastructure, but you should)
- use `[chunkhash]` in `output.chunkFilename` (Note that this example doesn't do this because of the example generator infrastructure, but you should)

example.js

```
_{{example.js}}_
```

webpack.config.js

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

index.html

```html
<html>
    <head> </head>
    <body>
        <!-- inlined minimized file "runtime~main.[chunkhash].js" -->
        <script>
            _{{production:dist/runtime~main.chunkhash.js}}_
        </script>

        <script src="dist/main.[chunkhash].js"></script>
    </body>
</html>
```

dist/runtime~main.[chunkhash].js

```
_{{dist/runtime~main.chunkhash.js}}_
```

dist/main.[chunkhash].js

`_{{dist/main.chunkhash.js}}_`

Info

Unoptimized

`_{{stdout}}_`

Production mode

`_{{production:stdout}}_`