Webpack

37

src/index.js

```
import bar from './bar';

bar();
```

src/bar.js

```
export default function bar() {
  //
}
```

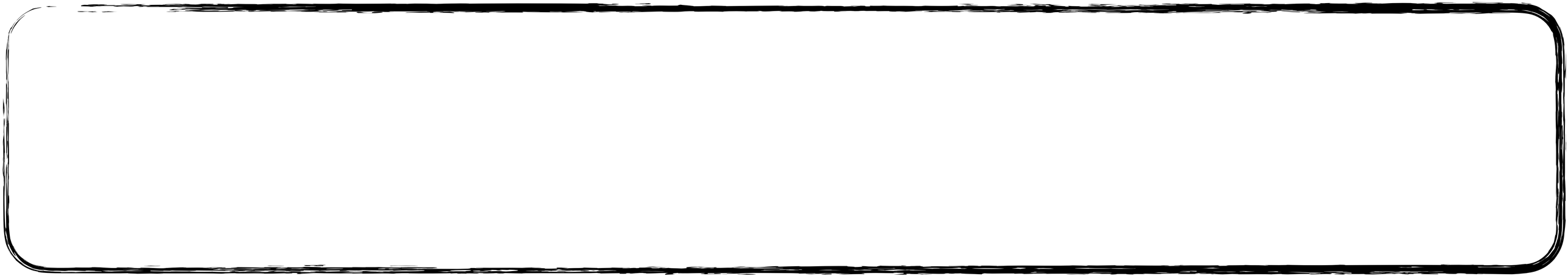
page.html

```
<!doctype html>
<html>
<head> ... </head>
<body> ...
    <script
        src="dist/bundle.js"></script>
</body>
</html>
```

webpack.config.js

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    path:
      path.resolve(__dirname,'dist'),
    filename: 'bundle.js'
  }
};
```



Source Code

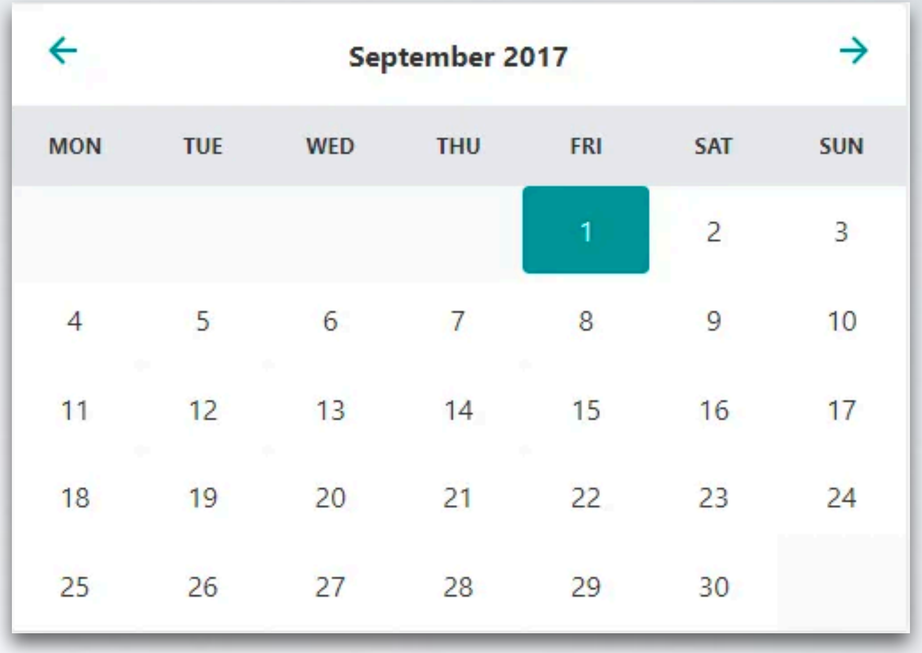
Entry Point

Config File

利用entry js，將關聯檔案全部合併(Bundle)成一個js並進行優化

元件化

- 有些前端框架甚至可以做到元件化
- 不只是物件化
  - 將html, css, js整個物件化起來
- 元件內部之html, css, js
  不會影響到外部
- 也可定義與外部js
  之input, output接口



元件化後之日曆
內部運作邏輯已經與外部隔開

Webpack

Entry Point

Source Code

src/index.js

```
import bar from './bar';

bar();
```

src/bar.js

```
export default function bar() {
  //
}
```

webpack.config.js

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    path:
      path.resolve(__dirname,'dist'),
    filename: 'bundle.js'
  }
};
```

Config File

page.html

```
<!doctype html>
<html>
<head> ... </head>
<body> ...
   <script
      src="dist/bundle.js"></script>
</body>
</html>
```

利用entry js，將關聯檔案全部合併(Bundle)成一個js並進行優化