34 - Utilizando Loaders no Webpack

1. Nessa aula continuaremos utilizando a pasta da aulas anteriores de webpack, mas faremos alguns rápidos ajustes nos arquivos. Vamos ajustar a configuração do webpack para que fique com apenas um ponto de entrada e que a saída volte a ser o padrão “dist/index.js”:

```
// webpack.config.js const path = require('path') module.exports = {
entry: { index: './src/index.js' }, mode: 'development' }
```

2. E o arquivo index.html na pasta “dist” deve incluir corretamente o script da saída do webpack:

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta
http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport"
content="width=device-width, initial-scale=1.0"> <title>Usando o
Webpack</title> <script src="index.js"></script> </head> <body>
<h1>Usando o Webpack</h1> </body> </html>
```

3. Exclua também a pasta public e quaisquer outros arquivos gerados anteriormente, deixando apenas o arquivo “dist/index.html” e o arquivo “src/index.js”.
4. Com tudo preparado, podemos começar. Vamos criar uma pasta “styles” dentro de “src” e dentro dela o arquivo “style.css” com o seguinte conteúdo:

   Obs.: Isso fará a nossa página ter um fundo escuro e um texto claro, apenas para conseguimos visualizar que o arquivo CSS foi carregado corretamente.

```
body { background-color: #232323; color: #efefef; }
```

5. Como eu expliquei anteriormente, os loaders permitem que nós transformemos os módulos, o que permite que o webpack também seja capaz de entender outros formatos de arquivo além do javascript e do JSON. Vamos instalar dois loaders que nos ajudarão a incluir arquivos CSS no webpack, o style-loader e o css-loader:

```
npm install --save-dev style-loader css-loader
```

6. Com esses dois pacotes instalados só precisamos configurar o webpack para utilizá-los:

```
module.exports = { entry: { index: './src/index.js' }, mode:
'development', module: { rules: [{ test: /\.css$/, use: ['style-
loader', 'css-loader'] }] } }
```

7. Com o webpack configurado precisamos incluir o arquivo “style.css” no nosso ponto de entrada. Isso é fundamental para que tudo funcione corretamente:

   Obs.: Repare que ao rodar agora o `npx webpack` o “bundle.min.js” continua sendo o único arquivo de saída, porém ao acessar a página no navegador ela já terá o fundo escuro.

```
import dayjs from 'dayjs' import './styles/style.css' alert(`Hoje é:
${dayjs().format("DD/MM/YYYY")}`)
```