

O **Webpack** é um *module bundler* — uma ferramenta que pega todos os arquivos e módulos da sua aplicação (JavaScript, CSS, imagens, fontes, etc.) e os transforma em **pacotes otimizados** para serem carregados pelo navegador.

Ele resolve três problemas principais no desenvolvimento moderno:

1. Agrupamento (Bundling)

Antigamente você precisava incluir vários `<script>` no HTML — um para cada arquivo JS.

Com Webpack, você importa tudo com `import` ou `require` e ele **gera um único arquivo final**, por exemplo:

```
/dist/bundle.js
```

Isso melhora carregamento, organização e evita conflitos.

2. Transpilar código moderno

Navegadores não entendem 100% de código moderno (ES6+, JSX, TypeScript).

O Webpack permite usar *loaders*, como:

- **babel-loader** → transforma ES6+ ou JSX em JS compatível
- **ts-loader** → para TypeScript
- **style-loader / css-loader** → para processar CSS

Exemplo de configuração:

```
module: {  
  rules: [  
    {  
      test: /\.jsx?$/,  
      use: 'babel-loader',  
      exclude: /node_modules/  
    }  
  ]  
}
```

```
}
```

3. Gerenciar assets (CSS, imagens, fontes)

Com Webpack você pode importar arquivos diretamente no JS:

```
import "./styles.css"
import logo from "./logo.png"
```

E o Webpack empacota tudo, otimizando:

- compressão
- minificação
- cache busting (gerar nomes únicos com hash)

4. Development Server

Com o `webpack-dev-server` você tem:

- hot reloading (atualização automática)
- servidor local rápido
- build em tempo real

Comando:

```
npx webpack serve
```

5.Exemplo completo de webpack.config.js

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

```
module.exports = {
  entry: './src/index.js',
  output: {
```

```
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
  module: {
    rules: [
      {
        test: /\.jsx?$/,
        exclude: /node_modules/,
        use: 'babel-loader'
      },
      {
        test: /\.css$/,
        use: ['style-loader', 'css-loader']
      }
    ]
  },
  devServer: {
    static: './dist',
    port: 3000
  }
};
```

6. Por que o React usa Webpack?

O React usa Webpack (diretamente ou via ferramentas como Vite / CRA) porque:

- JSX precisa ser transpiling pelo Babel
- importa CSS e imagens diretamente
- cria um bundle otimizado com tree-shaking
- permite ambiente de desenvolvimento mais rápido
- reduz o tamanho final do app.

7. Criando um projeto React + Webpack do zero (sem Vite, sem CRA)

7.1. Crie a pasta do projeto

```
mkdir react-webpack-app
```

```
cd react-webpack-app
```

7.2. Inicialize o `package.json`

```
npm init -y
```

7.3. Instale as dependências do React

```
npm install react react-dom
```

7.4. Instale Webpack e Babel (dependências de build)

```
npm install -D webpack webpack-cli webpack-dev-server
```

Instalar Babel:

```
npm install -D @babel/core @babel/preset-env @babel/preset-react  
babel-loader
```

Loaders para CSS (caso queira importar CSS):

```
npm install -D style-loader css-loader
```

7.5. Crie a estrutura de pastas

```
react-webpack-app/
```

```
├── /src
|   ├── index.js
|   ├── App.jsx
|   └── styles.css (opcional)
├── index.html
├── webpack.config.js
└── .babelrc
```

7.6. Arquivos do Projeto

index.html

Crie na raiz:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>React com Webpack</title>

</head>

<body>

  <div id="root"></div>

  <script src="./dist/bundle.js"></script>

</body>

</html>
```

✓ src/App.jsx

```
export default function App() {  
  return <h1>Hello React + Webpack!</h1>;  
}
```

✓ src/index.js

```
import React from "react";  
  
import { createRoot } from "react-dom/client";  
  
import App from "../App";  
  
const root = createRoot(document.getElementById("root"));  
  
root.render(<App />);
```

7.7. Configuração do Babel

Crie o arquivo `.babelrc`:

```
{  
  "presets": ["@babel/preset-env", "@babel/preset-react",  
    {"runtime": "automatic"}]  
}
```

7.8. Configuração do Webpack

Crie `webpack.config.js`:

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

```
module.exports = {  
  entry: "./src/index.js",  
  
  output: {  
    path: path.resolve(__dirname, "dist"),  
    filename: "bundle.js",  
  },  
  
  resolve: {  
    extensions: [".js", ".jsx"],  
  },  
  
  module: {  
    rules: [  
      {  
        test: /\. (js|jsx) $/,  
        exclude: /node_modules/,  
        use: "babel-loader",  
      },  
      {  
        test: /\.css$/,  
        use: ["style-loader", "css-loader"],  
      },  
    ],  
  },  
}
```

```
    },  
  
    devServer: {  
      static: {  
        directory: path.join(__dirname, "/"),  
      },  
      port: 3000,  
      open: true,  
      hot: true,  
    },  
  };
```

7.9. Scripts no package.json

No `package.json`, adicione:

```
{  
  "scripts": {  
    "dev": "webpack serve --mode development",  
    "build": "webpack --mode production"  
  }  
}
```

7.10. Rodar o projeto

Ambiente de desenvolvimento:

```
npm run dev
```

Abre em **`http://localhost:3000`**.

Build de produção:

```
npm run build
```

Gera a pasta **dist** com o bundle final.