報道各位

2020年8月19日
株式会社インプレスR&D
https://nextpublishing.jp/

そのプラグインってどうして必要なの！？

『React 環境設定の教科書』発行

技術の泉シリーズ、8月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『React 環境設定の教科書』(著者:井手 優太)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『React環境設定の教科書』
https://nextpublishing.jp/isbn/9784844378778

著者:井手 優太
小売希望価格:電子書籍版 1600円(税別)／印刷書籍版 2000円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5判／カラー／本文122ページ
ISBN:978-4-8443-7877-8
発行:インプレスR&D

＜＜発行主旨・内容紹介＞＞

本書はReactをテーマに、Webフロントエンド開発の環境構築を1つ1つ丁寧に解説します。特に設定が足りない状態で動かすとどうなるのか、なぜその設定が必要なのかについて踏み込んで説明しています。

環境構築の難しさは、設定をどれか1つでも間違えると動かないことにあります。そのため、環境構築をする際は1つ1つの設定で自分が何をしているのかを正確に把握しなければいけません。

この本では暗黙的に「そういうものだから」と思われがちな設定に注目し、解説します。どの設定を足さなければ動かないのか、サンプルコードを示しながら学ぶことができます。

(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

React をテーマに、環境構築を解説

リスト4.2: src/sub.js

```
export const hello = () => {
  console.log("hello");
};
```

バンドルします。

```
$ npx webpack
  Asset       Size  Chunks             Chunk Names
index.js  972 bytes       0  [emitted]  main
Entrypoint main = index.js
[0] ./src/index.js + 1 modules 97 bytes {0} [built]
    | ./src/index.js 41 bytes [built]
    | ./src/sub.js 56 bytes [built]
```

ここでindex.jsをentry pointとして読み込んだ際、sub.jsのconsole.log("hello");が生成物に含まれていると、依存解決が成功したことになります。書き出されたファイルは次の通りです(見やすくするために整形しています)。

リスト4.3: dist/main.js

```
!(function(e) {
  var t = {};

  省略,

  function(e, t, r) {
    "use strict";
    r.r(t);
    console.log("hello");
  })
```

ちゃんとconsole.log("hello");が組み込まれていることが確認できました。つまり、依存解決に成功しています。

webpackはv4からzero configでの実行も可能で、設定ファイルを書かなければビルドターゲットがsrc/index.jsで出力先をdist/main.jsとし、諸々の最適化を加えて出力します。そのため、設定ファイルを指定せずともビルドできました。もちろん、設定ファイルを活用した方が、差分管理や便利な機能を導入する上で好ましいため、設定ファイルについて次の節で見ていきます。

4.2 webpackの設定ファイル

wepbackは、設定ファイルとしてwebpack.config.jsを見ます。

リスト4.4: webpack.config.js

```
const HtmlWebpackPlugin = require("html-webpack-plugin");
const Dotenv = require("dotenv-webpack");
const path = require("path");

module.exports = {
  mode: process.env.NODE_ENV,
  entry: "./src/main.tsx",
  output: {
    path: path.resolve(__dirname, "./dist"),
    filename: "build.js",
  },
  module: {
    rules: [
      {
        test: /\.(ts|tsx)$/,
        use: [
          {
            loader: "ts-loader",
          },
        ],
      },
    ],
  },
  resolve: {
    extensions: [".js", ".ts", ".tsx", ".css"],
  },
  plugins: [
    new HtmlWebpackPlugin(),
    new Dotenv(),
  ],
};
```

その設定ファイル内で指定できる設定は、次の通りです。

4.2.1 mode

ビルド時の最適化について設定できます。取りうる値は、none、development、productionです。指定しなかった場合のデフォルト値は、productionです。productionは最適化をしてくれ、noneは何も最適化を行いません。developmentを設定すると、Source Map[6]を有効にできます。

6. ビルド後のソースとビルド前のソースを関連付けるファイル。これを用いれば、webpackでビルドしたファイルのデバッグが容易になる。

暗黙的に「そういうものだ」と思われがちな設定も詳説

リスト8.19: テスト

```
expect(getByTestId("clicked-status")).toHaveTextContent("on");
```

テストコードを実行すると、このようになります。

リスト8.20: テストに成功

```
$ npx jest
 PASS  src/main.test.tsx
  ✓ mounted text (34ms)
  ✓ click once (18ms)
  ✓ click twice (7ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.679s
Ran all test suites.
  Done in 5.17s.
```

成功しました。全体としては、このようなコードになります。

リスト8.21: テストコード

```
import React from "react";
import { render, fireEvent } from "@testing-library/react";
import "@testing-library/jest-dom/extend-expect";
import App from "./App";

it("mounted text", () => {
  const { getByTestId } = render(<App></App>);
  expect(getByTestId("text")).toHaveTextContent("mounted text");
});

it("click once", () => {
  const { getByTestId } = render(<App></App>);
  fireEvent(
    getByTestId("btn"),
    new MouseEvent("click", {
      bubbles: true,
      cancelable: true
    })
  );
  expect(getByTestId("clicked-status")).toHaveTextContent("on");
});

it("click twice", () => {
  const { getByTestId } = render(<App></App>);
  fireEvent(
    getByTestId("btn"),
    new MouseEvent("click", {
      bubbles: true,
      cancelable: true
    })
  );
  fireEvent(
    getByTestId("btn"),
    new MouseEvent("click", {
      bubbles: true,
      cancelable: true
    })
  );
  expect(getByTestId("clicked-status")).toHaveTextContent("off");
});
```

@testing-library/reactを利用すると、「ユーザーがこのような操作をしたときに画面はこうなる」といったことをテストできるため、シナリオベースでのテストがしやすいです。UIのテストは難しいとされますが、シナリオベースでのテストは比較的書きやすいです。

8.3 Jestで設定する項目

Jestの設定項目は多岐に渡ります。そのすべてを説明することは難しく、その必要もありません。そこで、重要な概念とテストが実行できないときの原因になりやすい項目について挙げます。

8.3.1 transform

transformerとの対応を宣言できます。transformerとは、ソースコードを変換する同期的な関数です。平たくいうと、トランスパイラとそれに対応するファイルパスを指定できます。

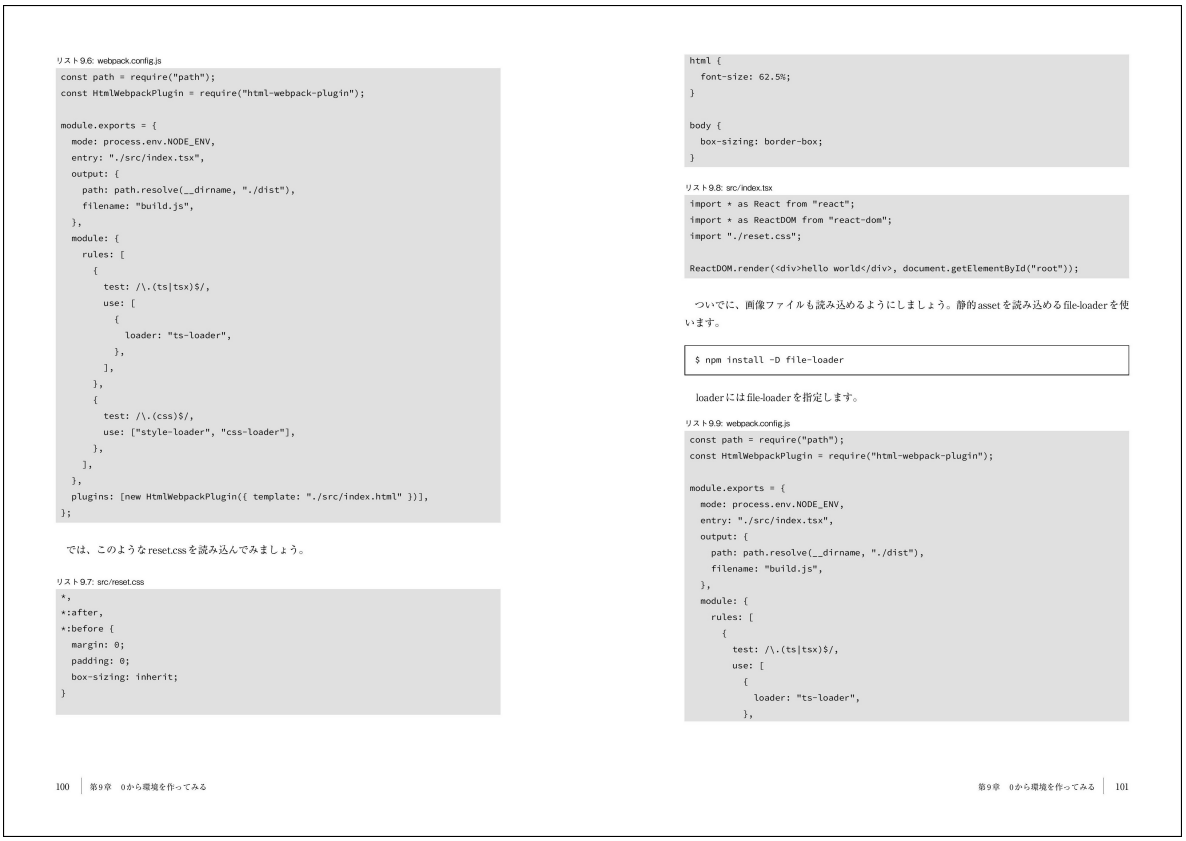
たとえば、次のように指定できます。

リスト8.22: jest.config.js

```
...,

"transform": {
  "^.+\\.[t|j]sx?$": "babel-jest"
},
```

サンプルコードを示しながら学習



リスト9.6: webpack.config.js

```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  mode: process.env.NODE_ENV,
  entry: "./src/index.tsx",
  output: {
    path: path.resolve(__dirname, "./dist"),
    filename: "build.js",
  },
  module: {
    rules: [
      {
        test: /\.(ts|tsx)$/,
        use: [
          {
            loader: "ts-loader",
          },
        ],
      },
      {
        test: /\.(css)$/,
        use: ["style-loader", "css-loader"],
      },
    ],
  },
  plugins: [new HtmlWebpackPlugin({ template: "./src/index.html" })],
};
```

では、このようなreset.cssを読み込んでみましょう。

リスト9.7: src/reset.css

```
*,
*:after,
*:before {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}
```

100 | 第9章　0から環境を作ってみる

```
html {
  font-size: 62.5%;
}

body {
  box-sizing: border-box;
}
```

リスト9.8: src/index.tsx

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import "./reset.css";

ReactDOM.render(<div>hello world</div>, document.getElementById("root"));
```

ついでに、画像ファイルも読み込めるようにしましょう。静的assetを読み込めるfile-loaderを使います。

```
$ npm install -D file-loader
```

loaderにはfile-loaderを指定します。

リスト9.9: webpack.config.js

```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  mode: process.env.NODE_ENV,
  entry: "./src/index.tsx",
  output: {
    path: path.resolve(__dirname, "./dist"),
    filename: "build.js",
  },
  module: {
    rules: [
      {
        test: /\.(ts|tsx)$/,
        use: [
          {
            loader: "ts-loader",
          },
```

第9章　0から環境を作ってみる | 101

＜＜目次＞＞

＜＜著者紹介＞＞

約 3 年間、大手事業会社にて営業や開発を経験し、現在はフリーランスとして独立。2019 年 2 月から株式会社 TechBowl が提供する TechTrain でメンターを始める。2019 年 7 月には株式会社 Kaizen Platform にて社内横断のフロントエンド開発基盤・テスト基盤の設計・開発・保守を担当する。現在は新アーキテクチャ移行に向けたテストやドキュメントの整備、パッケージガーデニングとその調査に従事している。

＜＜販売ストア＞＞

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、
honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。

※　全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/

株式会社インプレスR&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：松本大輔、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「モバイルサービス」「学術・理工学」「旅・鉄道」を主要テーマに専門性の高いメディア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp