Webデベロッパーのための

React開発入門

JavaScript UIライブラリの基本と活用

柴田 文彦[著]

イチからわかる!
Reactの仕組みと使い方

Reactのコンセプト、コンポーネント、JSX、
活用テクニック、一歩進んだ使い方を解説!

UIコードの
再利用化と
速度向上
を図る!

インプレス

■サンプルコードのダウンロード／正誤表のWebページ
本書のサンプルコードは以下のURLのページからダウンロードできます。また、正誤表を掲載した場合はこのページに表示されます。

http://book.impress.co.jp/books/1116101049

◎本文中に登場する会社名、製品名、サービス名は、各社の登録商標または商標です。

◎本書の内容は本書執筆時点のものです。本書で紹介した製品／サービスなどの名前や内容は変更される可能性があります。

◎本書の内容に基づく実施・運用において発生したいかなる損害も、著者ならびに株式会社インプレスは一切の責任を負いません。

◎本文中では®、TM、©マークは明記しておりません。

はじめに

その対象がどんなものであれ、プログラミングは楽しいものです。楽しくなければ学ぶ意欲も起こらず、長く続けることもできません。プログラミングが楽しいものであるためには、自分の行為についてよく理解していることが必要です。また、プログラムが自分の思った通りに動き、その結果として何らかの成果が、願わくば目覚ましい効果が実感できるものでなければなりません。

この本は、Reactを利用したWebアプリのプログラミング入門書です。それに関する基礎的な知識、経験、そして狙った成果を得るための技術を習得することを目指しています。つまり、その分野のプログラミングが楽しいと感じられるようになりたい人のための本です。

この本を書くにあたっては、読者としてある程度のレベルの人を想定しています。簡単に言えば、HTML、CSS、JavaScriptに関して基礎的な知識を持っている人です。それらを使ったシンプルなWebページを、テキストエディタだけを使って書いたことがある人なら申し分ないでしょう。もちろん、それほど高度な知識や経験を想定しているわけではありません。本書の中では、HTMLやJavaScriptそのものについては特に解説していませんが、入門レベルを超えると思われる事項については、折に触れて説明を加えています。そして、Reactの真髄を学ぶ上で、基礎知識の不足が障害にならないように心がけています。

この本を使ってReactのプログラミングについて学ぶためには、Webブラウザが1本あれば十分です。自前のサーバーを立てたりする必要もありません。バージョンさえ新しければ、タブレット上のものでも大丈夫でしょう。装備や環境の整備に気を散らすことなく、本題に集中してプログラミングを楽しんでください。

―――――― 2016年11月　柴田文彦

c o n t e n t s

Chapter 1 Reactとは？　1

1-1 Reactの概要と特長｜2

Reactは一種のJavaScriptライブラリ？――2
Declarative――3
Component-Based――4
Learn Once, Write Anywhere――5

1-2 Reactを使うために必要な準備｜6

さまざまな動作環境――6

1-3 Reactを動かしてみよう｜9

JSFiddle上での試運転――9

1-4 Reactを理解する5つのキーワード｜13

構成要素で考えるReact――13
コンポーネント――13
ステート――14
プロパティ――14
JSX――15
仮想DOM――15

Chapter 2 Reactのコンポーネント　17

2-1 コンポーネント定義の基本｜18

HTML要素をコンポーネント化する――18
要素の属性を指定する――21
カスタムコンポーネントを作る――23

2-2 内容が変化するコンポーネント｜27

文字列を受け取って表示するコンポーネント――27
与えた式を計算するコンポーネント――30

2-3 ユーザーからの入力を受け付けるコンポーネント｜33

1つのコンポーネント内に複数の要素を配置する――33
文字列を入力するinput要素を配置する――36
ユーザー入力に反応する仕組みを実現する――37

2-4 コンポーネントのライフサイクル｜42

ライフサイクルメソッド――42
典型的なライフサイクルシーケンス――47
ライフサイクルの変遷を観察する――49

Chapter 3 JSXの基本　53

3-1 JSXの概要とメリット｜54

JSXを使うには――54
JSXとBabel――55
JSXとHTMLとの違い――58

3-2 JSXの中で使うJavaScript｜61

JSXでカスタムコンポーネントを記述する――61
JSXで複数の要素を配置する――63
JSXでステート付きのコンポーネントを作成する――64
条件（三項）演算を使う――66
真偽値の属性を指定する――69
JSXの中のコメントの書き方――71

3-3 スプレッド属性を利用する｜75

スプレッド属性とは――75
スプレッド属性でスタイルを設定する――79
カスタムコンポーネントでスプレッド属性を利用する――80
デフォルトの属性をスプレッド属性で設定する――83
デフォルトの属性をオーバーライドする――86

3-4 JSXトランスパイラの使い方｜89

Webサイト上のトランスパイラを利用する――89
ブラウザ用のトランスパイラを利用する――92
JSXを別のファイルに保存して動かす――95

Chapter 4 Reactを使いこなす　101

4-1 Reactで使うフォーム｜102

HTMLとの比較――102
コントロールドコンポーネント――106
type属性とchecked属性でラジオボタンを実現――110
type属性で複数選択可能なチェックボックスを実現――117
プロパティによって選択状態の初期値を設定する――122
value属性でセレクトボックスを選択可能にする――125
TextAreaコンポーネントではvalue属性で値を設定――130

4-2 フォームのサブミットとバリデーション｜133

formタグとonSubmit属性――133
ステート値のバリデーション――134

4-3 動的に作成するコンポーネント｜138

HTMLとReactのtableタグの違い――138
配列要素を自動的に展開して表示するmapメソッド――144

4-4 コンポーネント間の機能の共有｜149

マルチコンポーネント化――149
ネームスペースドコンポーネント――153
Mixinの利用――157

Chapter 5 Reactの一歩進んだ使い方　167

5-1　アドオンによる機能拡張｜168

Reactのアドオンについて――168
公式なアドオン――168
Reactアドオンを使う準備――171
アドオンの使用例――171

5-2　アニメーションの基本テクニック｜177

Reactアニメーションの種類――177
CSSアニメーションを確認する――177
Reactアドオンによる出し入れのアニメーション――182
Reactアドオンによる初期マウントのアニメーション――190
トランスフォームを使ったアニメーション――194
ReactアニメーションをつかったToDoリスト――197

5-3　Reactの開発ツール｜207

React純正のブラウザ拡張機能――207
React Developer Toolsの簡単な使い方――211

5-4　Reactの周辺技術｜218

次に何を学ぶべきか？――218
Flux――218
React Native――220
いくつかのUIライブラリ――222

索引――226
著者プロフィール――231

01

Reactとは?

本書の最初の章では、「Reactとは何か?」という話から始めます。それからReactを使うための準備について述べ、少しだけですが実際にReactを使ってみるところまで進みます。また実際に動かしてみた後で、Reactを理解するための5つのキーワードについて簡潔に説明します。それだけでReactが何か、どのように動いているかが理解できるわけではありませんが、本書を読み進めるための予備知識を得ることはできるはずです。本書を読み終わった後で、これらのキーワードがReactを理解するために、文字通りキーとなる用語だったと納得されるでしょう。

もしすでにReactについて、ある程度の予備知識があり、早く実際のプログラミング方法を知りたいという方は、この章を飛ばして、2章から読み始めてもかまいません。

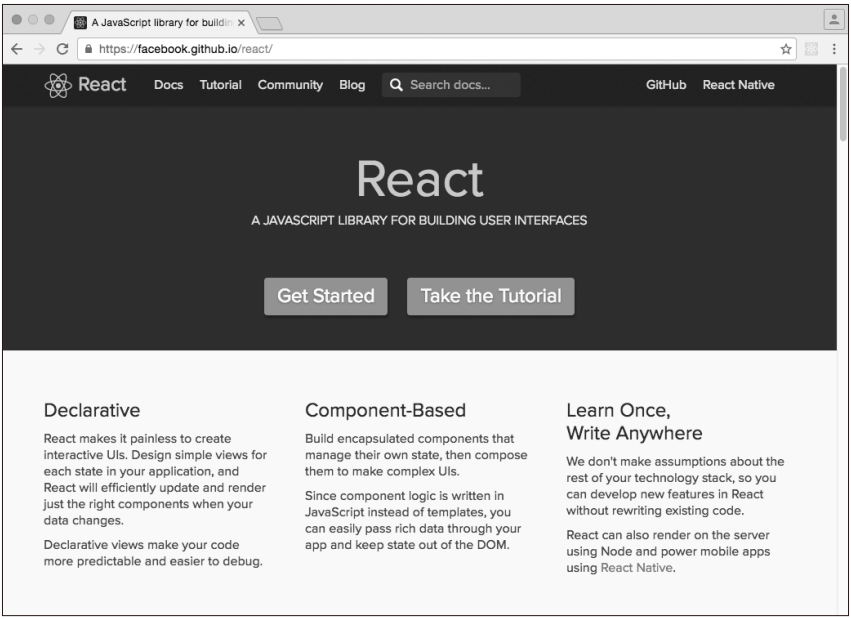
1-1

Reactの概要と特長

▶▶▶▶Reactは一種のJavaScriptライブラリ？

Reactとは何か？──それに対する答えは、見る角度によってさまざまなものが考えられます。しかし、1つの簡潔で明確な答えは、Reactの公式サイト（https://facebook.github.io/react/）にはっきりと掲示されています（図1-1）。それによれば、Reactは「A JavaScript Library For Building User Interfaces」であるとされています。日本語に訳せば、「ユーザーインターフェースを構築するためのJavaScriptライブラリの1つ」ということになるでしょう。

▶▶ 図1-1：React公式サイト。一通りのドキュメント、チュートリアル、最新情報を発信するブログが集約されている



これは疑問の余地も、追加の説明の必要もほとんど感じられない簡明なものです。と言いつつ、いきなり余談ですが、ReactはJavaScriptの「フレームワーク」だと思っていた人も少なくないかもしれません。それは言葉の定義、解釈の問題であって、ライブラリでもフレームワークでもどちらでもよいと思われるかもしれません。それでもReactの開発元としては、フレームワークではなく、ライブラリという言葉を選んでいるのは事実です。ライブラリとフレームワークは何が違うのか、結局同じものを指しているのではないか、といった疑問はしばしば耳にします。一般的に、ソフトウェアの規模としては、ライブラリよりもフレームワークのほうが大きいという印象があります。フレームワークは、複数のライブラリや、それに属するリソース、あるいはユーティリティのようなものも含む大規模なものという考え方もあります。もちろん異論もあるでしょう。ライブラリとフレームワークの違いは規模の大きさには関係ないという意見も聞こえてきそうです。それでも、ライブラリのほうがフレームワークよりも大掛かりなソフトウェアを指すと思っている人は少ないのではないでしょうか。それを考えると、Reactがライブラリを名乗っているのには、他の一般的なJavaScriptによるUIフレームワークよりもコンパクトなものであるという主張が含まれているように感じられます。

それはともかくとして、Reactの公式サイトには、図1-1にも見えるように、Reactの定義の下に、Reactの特徴を表現していると思われる3つのキーワード、あるいはフレーズが挙げられ、それぞれに簡単な説明があります。その3つとは、「Declarative」「Component-Based」「Learn Once, Write Anywhere」です。それぞれについて見ていきましょう。

▶▶▶▶Declarative

「Declarative」は、日本語に訳せば「宣言的な」あるいは「宣言型の」ということになるでしょう。プログラミング言語そのものに興味があるような人なら、それを聞いただけでだいたいの特徴が把握できるのではないでしょうか。

Declarativeが意味していることを理解するには、その反意語と比べて考えるのが効果的でしょう。プログラミングの世界でDeclarativeの反意語は「Procedural」です。日本語では「手続き的な」あるいは「手続き型の」という意味になります。

プログラミングで使う言語には、大別して宣言的なものと手続き的なものの2種類があることになっています。Webの世界で言えば、宣言的な言語の代表的なものはHTMLです。HTMLでは、基本的に要素とその属性を延々と置いていくだけで、何をどうするといった手続き的な記述は含みません。

一方、Webで手続き的なものの代表はJavaScriptです。これは、一般的なプログラミング言語なので、何を、どういう場合に、どうする、といった手順を主に記述することになります。

ReactはJavaScriptのライブラリなのに手続き的ではなく宣言的であるところが最大の特徴の1つです。HTMLで言えば要素、Reactではコンポーネントと呼ぶものを宣言として置いておくと、アプリ内の状態に応じて、Reactが各コンポーネント（要素）の見え方、表示内容を、必要に応じて自動的にアップデートしてくれます。

この特徴のおかげで、ReactはJavaScriptのプログラマーではなく、Webデザイナーと呼ばれるような人たちでも利用しやすくなっています。もちろんプログラマーにとってもメリットは大きく、見通しのよいプログラムを書くことを可能にしてくれます。

▶▶▶▶Component-Based

「Component-Based」は、一般的には「コンポーネントベースの」とか、「コンポーネントを基本とする」などと訳されるでしょう。Reactがコンポーネントというものを基本としているという意味で、もちろん間違いではありません。しかし、なんとかbasedには、もう1つのもっと具体的な意味があります。それは「主原料が～である」ということです。こう考えると、Reactの特徴がよりはっきりと理解できます。つまり、Reactはコンポーネントを主原料として、それらを組み合わせることで、複雑なユーザーインターフェースを構築するものなのです。コンポーネントは、それぞれ独立した部品として機能し、組み合わせることで有機的に作用するといったところでしょうか。

コンポーネント自体は前の特徴で述べたように宣言的に記述できます。また、複数のコンポーネントを組み合わせるための記述も宣言的です。HTMLに慣れた人なら、何の違和感もなくReactのコンポーネントを組み合わせてUIを構

成することができるでしょう。しかし、コンポーネント内のロジック部分、言い換えればユーザーの操作に対して応答する部分はJavaScriptで手続き的に記述することができます。それぞれの言語の特徴と適性を活かして、応答性の高いUIを効率的に実現することができるのです。

▶▶▶▶Learn Once, Write Anywhere

「Learn Once, Write Anywhere」は、日本語にすれば「一度学べば、どこにでも書ける」といった意味です。これは、明らかに「Write Once, Run Anywhere」のもじりです。後者は「一度書けば、どこででも動く」という意味で、1990年代にJava言語が登場したときのスローガンでした。Javaという1つの言語でプログラムを記述すれば、さまざまなデバイス上に用意されたJava仮想マシン上で、そのまま動作するという意味です。

Reactの場合も、Webに限って言えば、Javaと同じように、一度書けばどこででも動く、ということが言えるでしょう。しかし、Reactの動作環境は実はWebだけではありません。Reactから派生したフレームワーク「React Native」を使えば、iOSやAndroidのモバイルデバイス上でネイティブに動作するアプリケーションを作成することができます。React Nativeも基本的にはReactでアプリ画面を記述しますが、WebとiOSとAndroidで少しずつ記述が異なります。しかし、基本は同じなので、一度（Reactを）学べば、どこ（Web、iOS、Android）にでも（プログラムが）書けます。

なお、React Nativeについては本書の5章で、少しだけ紹介しています。

いずれにしてもReactは、最近のネイティブアプリと見間違えるような動作を見せるWebアプリのUIを開発するために設計されたライブラリです。開発元はFacebookで、実際にFacebookや、今やその傘下のInstagramでは、Web版の公式アプリを開発するためにReactを全面的に採用しています。それらはWebアプリ開発者にとっては最高の目標であり手本と言えるでしょう。そしてそれらを開発するためのツールは、Reactとして誰でも使えるように公開されているのです。逆に言えば、Reactを使えば、FacebookやInstagramと同等のWebアプリができるはずなのです。もちろんそれは容易ではありませんが、必要な条件の1つはReactを利用することによって整うのです。

1-2

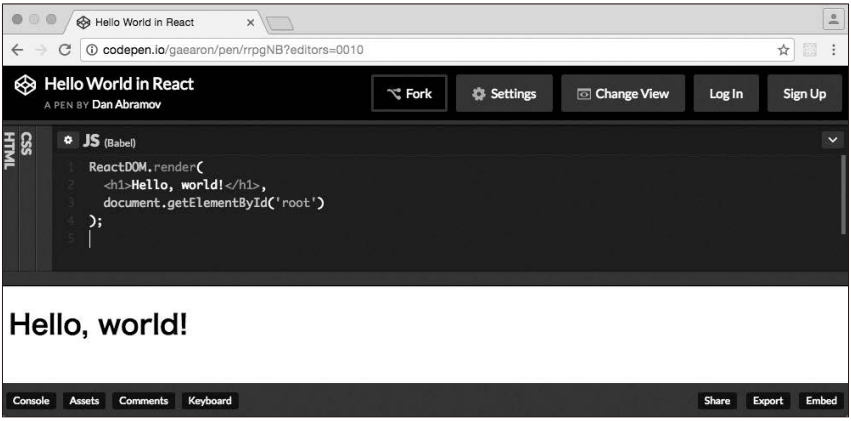
Reactを使うために必要な準備

▶▶▶▶さまざまな動作環境

Reactの動作環境は非常に幅広く、さまざまなバリエーションが考えられます。その動作環境のバリエーションの広さのおかげで、Reactを使う目的やアプリの規模などの幅広いバリエーションに対応できるのです。

簡単にReactの動作を確かめるだけなら、Webブラウザが1本あればこと足ります。ローカルにファイルを保存したり、環境を整える必要もないので、モバイルデバイスのWebブラウザでも大丈夫です。これは、CodePenや、JS Bin、JSFiddleといったクラウド上のWebアプリテスト環境のようなものを利用することで可能となります。Reactの公式サイトでも、CodePenを使ったHello Worldを公開しています（http://codepen.io/gaearon/pen/rrpgNB?editors=0010）（図1-2）。

▶▶ 図1-2：Reactの公式サイトからリンクされたCodePenのページでは、ReactのHello Worldが動作する



この方法は、主に開発の初期段階で他人と共同開発したり、コードについて意見を求めたり、作成したアプリの動作、アイディアを評価してもらったりするのにも有効です。ただし、これだけで本格的なWebアプリを開発し、それを多くの人に実用的に使ってもらうには無理があるかもしれません。

次に手軽に使えるのは、ローカルのブラウザ上でReactを動かす方法です。そのためには、React本体のスクリプトをクラウド上のコードサーバーであるCDN（Content Delivery Network）の1つからダウンロードしてインクルードするか（読み込むようにするか）、あらかじめダウンロードしたReact本体を、所定の場所に保存しておき、そこからインクルードして動作させます。

これはある意味完結した動作環境であり、その気になればこうして開発したコードをWebサーバー上にデプロイして、一般公開することも可能です。ただし、この方法では、特にWebアプリの起動時に時間がかかります。そのためReactでは、このような方法で製品版を作ることは避けるように勧めています。

一方でReactの公式サイトでは、この方法で動作するReactのHello Worldのサンプルも公開しています。以下にその内容をそのまま示します（リスト1-1）。まだこの内容については説明しません。これは1つのHTMLファイルであり、すべて完結しています。そのファイル名を「index.html」などとしてローカルに保存し、適当なWebブラウザで開けば、ブラウザウィンドウに「Hello, world!」と表示されるはずです。

▶▶ リスト1-1：HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
    <script src="https://unpkg.com/react@latest/dist/react.js"></
script>
    <script src="https://unpkg.com/react-dom@latest/dist/react-
dom.js"></script>
    <script src="https://unpkg.com/babel-standalone@6.15.0/babel.
min.js"></script>
  </head>
```

```
  <body>
    <div id="root"></div>
    <script type="text/babel">

      ReactDOM.render(
        <h1>Hello, world!</h1>,
        document.getElementById('root')
      );

    </script>
  </body>
</html>
```

もう1つの方法は、サーバー上でパッケージ管理ツールnpmを使ってNode.jsや、React関連の必要なパッケージをインストールし、その上でReactとそれを利用するWebアプリを動かすものです。この方法を使えば、最終的な製品版まで開発して公開することが可能となります。ただし、開発段階でもサーバーが必要になるため、効率的な開発のためには開発マシン上にローカルなWebサーバーをインストールして、その上でデバッグしたりすることになります。普段から同様の方法でWebアプリを開発している人は別として、どうしても準備が大がかりになり、使用しているOS、マシン、あるいは目的によっても使用するパッケージや設定方法が異なってきます。そのため、あくまでReactの入門という位置付けの本書では、この方法は取り上げないことにします。React公式サイトのInstallationのページ（https://facebook.github.io/react/docs/installation.html）にも、この方法の1つのバリエーションについて簡単な説明があるので、必要に応じてそちらを参照してください。

1-3

Reactを動かしてみよう

▶▶▶▶JSFiddle上での試運転

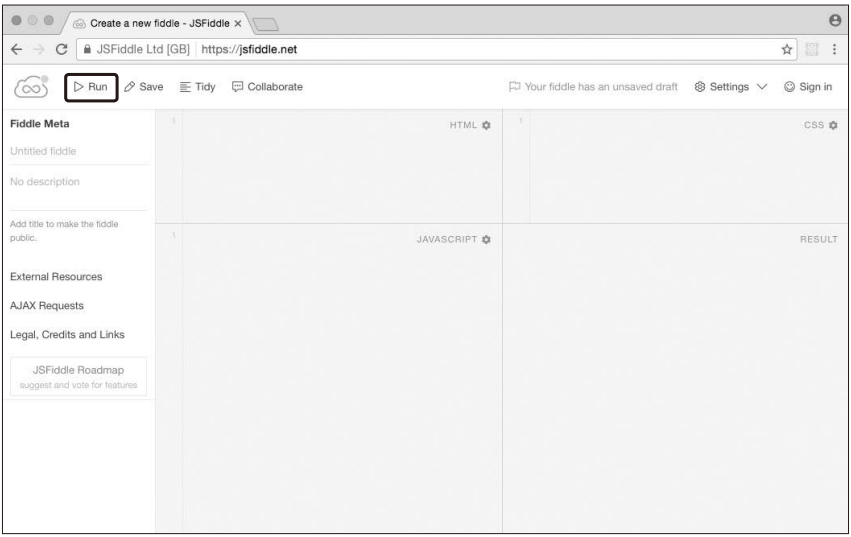
Reactの公式サイトから1クリックでCodePenのページを開いたり、ダウンロードしたHTMLファイルをブラウザで開いただけでは、自分でReactを動かしてみたという実感は得られないでしょう。そこで、環境はともかくとして、白紙の状態からコードを打ち込んで、Reactを利用したページが表示されるまでの過程を体験してみることにします。

ここでは、1-2節の「さまざまな動作環境」でも紹介したクラウド上のWebアプリテスト環境を使います。テスト環境として選んだのはJSFiddleですが、ほかに使い慣れた環境があれば、それでもかまいません。むしろ、本書に示した方法そのままではなく、読者の好みや主義で選んだ方法で試してみるほうが効果は大きいでしょう。JSFiddleを選んだ理由は、HTMLとJavaScriptのコードが縦に並んで見やすいという理由によります。ここしばらくは、HTMLとJavaScriptのみに注目すればよいからです。

なお、ブラウザとしては、Chromeを選びました。最新のものであれば、FirefoxやSafari、そのほかのものでもかまいません。本書では今後も一貫してChromeを使っていますが、別のブラウザでも試してみるとよいでしょう。

まずは、ChromeでJSFiddleのサイト（https://jsfiddle.net/）を開きます（図1-3）。JSFiddleでは、無料でユーザー登録も可能です。登録したユーザー名でログインしてから使えば、テスト中のソースコードを保存したり、他のユーザーと共有したりすることも可能です。ここではとりあえず、一種のHello Worldを試すだけなので、ログインせずに使っていきます。

▶▶ 図1-3：ChromeでJSFiddle（https://jsfiddle.net/）を開く



JSFiddleでは、ページ内の編集領域が田の字状に4つに分かれています。左上がHTML、左下がJavaScriptのソースコードを打ち込んで編集できるエディタになっています。右上は同様にCSSの編集領域ですが、とりあえずここは使いません。

右下は、これらのソースコードの結果として表示されるページのイメージです。左上の「▷ Run」のボタンをクリックするたびに更新されます。

まずは、HTMLから入力していきましょう。URLをタイプするのがやや面倒かもしれませんが、次の実質3行を左上のHTML領域に入力してください（リスト1-2）。

▶▶ リスト1-2：HTML

```
<script src="https://fb.me/react-15.3.0.js"></script>
<script src="https://fb.me/react-dom-15.3.0.js"></script>

<div id="content"></div>
```

この結果、HTMLが文法に従ってカラーリングされて表示されます（図1-4）。1行が長くなって領域の境界線にかかり、折り返されてしまう場合は、境界線をドラッグして領域の広さを調整することができます。

▶▶ 図1-4：まずHTMLを左上の領域に入力する

JSFiddleでは、お決まりのHTMLタグの入力を省くことができます。たとえば、<!DOCTYPE html>、<html>、<head>、<body>などは省略して問題ありません。また、JavaScriptのコードを外部スクリプトとして参照するための記述も不要です。

この状態ではまだ「▷ Run」をクリックしても、何も起こりません。続いて、左下の領域にJavaScriptのコードを入力しましょう（リスト1-3）。見かけは4行ですが、意味的には1つのメソッドの呼び出しの1行だけです。

▶▶ リスト1-3：JavaScript

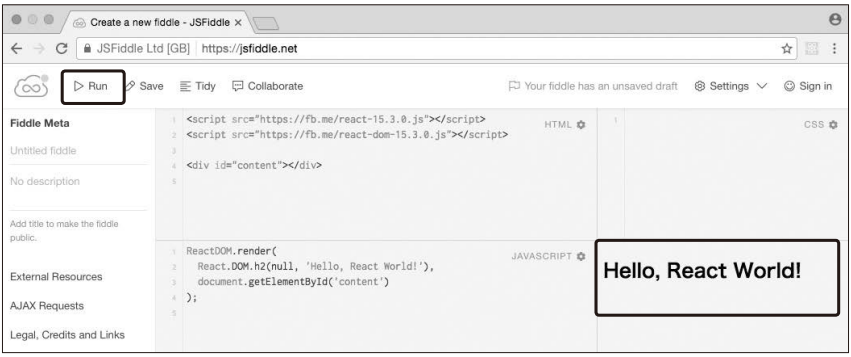
```
ReactDOM.render(
  React.DOM.h2(null, 'Hello, React World!'),
  document.getElementById('content')
);
```

このコードの意味については2章で説明しますが、なんとなくHTMLのcontentというIDの付いた要素の部分に「Hello, React World!」と表示するものであることはわかるでしょう（図1-5）。

▶▶ 図1-5：続いてJavaScriptを左下の領域に入力する

ここまできたら、後は「▷ Run」をクリックするだけです。その結果は、右下の領域にh2要素の太めの文字で「Hello, React World!」と表示されるはずです（図1-6）。

▶▶ 図1-6：「▷ Run」ボタンをクリックして結果を確認する



もし何も表示されない場合は、もう一度コードを見直すか、ブラウザのJavaScriptコンソールを表示して、エラーメッセージを確認して対処してください。

1-4

Reactを理解する5つのキーワード

▶▶▶▶構成要素で考えるReact

Reactがどのような特徴を備えたJavaScriptライブラリであるかは、すでにこの章の最初で述べました。しかし、それだけではReactがどのようなものであるかをイメージとしてつかむことは難しいでしょう。何のイメージもないまま、やみくもに新しい概念を学ぶのは困難です。

そこでここでは、少しでも的確なReactのイメージに近づけるように、Reactにまつわる5つのキーワードを取り上げて、簡単にそれらの意味を説明しておきます。これらはReactの構成要素、あるいはそれを支えるコンセプトのようなものと考えられます。以下の説明は、本書の2章以降を読み進めるための、基本的な用語解説の役割も果たします。ここだけで、それらの意味を完全に把握するのは無理ですし、そうする必要もありません。ただ、Reactのイメージをなんとなくでもつかむための助けと考えてください。

▶▶▶▶コンポーネント

コンポーネント（Component）については、すでに述べたように、Reactの公式サイトにReactの大きな特徴の1つとして「Component-Based」が掲げられています。それは「コンポーネントを主原料とする」と訳すと意味がわかりやすいということも先に述べた通りです。

いずれにしても、コンポーネントはReactの根幹をなすものであることは間違いありません。ReactによってWebアプリ、その中のページを構成するということは、コンポーネントを定義し、それを配置していくことにほかなりません。これを簡単に言えば、HTMLのエレメント＝要素を自由に拡張し、その中身をJavaScript

で記述し、HTMLの要素と同様に、あるいはそれらと混在させて配置していくということになるでしょう。コンポーネントは、作るときにはJavaScriptのメソッド、あるいはファンクションのようなものですが、使うときにはHTMLの要素のようなものと考えることもできます。

コンポーネントの実際の記述方法、使用方法については、2章以降でじっくりと段階を追って説明していきます。

▶▶▶▶ステート

ステート（State）は、コンポーネントの内部の変数のようなものです。ステートは、プログラムの実行中、コンポーネントが生成される際に初期化され、その後はユーザーによる操作の結果など、コンポーネントの状態の変化を保持するために使われます。ステートはコンポーネントの外部からはアクセスできません。ただし、初期化の際には、次に説明するプロパティを外部から与えることで、ステートの値に反映させることは可能です。

ステートは、非常に重要な働きをする反面、数が多くなってくると、コンポーネントが複雑になり、人間がその状態を把握するのが難しくなります。Reactでは、できるだけステートに頼らないコンポーネントの設計を勧めていて、使わないで済むなら、使わないほうがよいという意見もあります。

▶▶▶▶プロパティ

プロパティ（Props）は、コンポーネントに外部から与えられるオプションのようなものです。HTMLで言えば、要素を配置する際にタグ名の後に与える属性に相当します。実際にReactのコンポーネントを配置する際には、<>で囲むタグの中にタグ名、つまりコンポーネント名に続いて「プロパティの名前＝値」のように記述します。

コンポーネントの内部では、プロパティはステートと同じように変数としてアクセスできます。ただし、ステートと違ってコンポーネント自身はプロパティの値を変更してはいけません。プロパティは外の世界から一方的に与えられるものであって、それを使って外の世界に何かを伝えることはできないのです。

▶▶▶▶JSX

JSXの先頭の「JS」は、JavaScriptを表しています。最後の「X」はXMLのXです。これはJavaScriptの中で、XMLの文法によって、Reactのコンポーネントを配置できるようにするための一種の言語です。これにより、JavaScriptとHTMLを混在させるような感覚で、WebページのUIを記述できるようになります。

JSXはReactとは独立して存在する言語であって、ReactがJSXに依存しているわけではありません。JSXについて好きになれなければ、JSXをまったく使わずにReactコンポーネントを記述することができます。その場合でも、機能的にはReactの能力が損なわれることはありません。しかし、簡潔で直接的な表現という点ではReactならではの魅力は損なわれるように思われます。

JSXは、JavaScript部分は標準的なJavaScriptですが、その中にXMLによる記述が混在するので、そのままではWebブラウザで実行することはできません。そのため、実行前に何らかの方法でJavaScriptに変換する必要があります。それについては3章で取り上げています。

▶▶▶▶仮想DOM

仮想DOM（Virtual DOM）は、コンポーネントと並んで、Reactの中核をなす特徴的な技術です。コンポーネントが外面的なものとすれば、仮想DOMは内面的なものと考えられます。Reactのコンポーネントを記述する際には、特に仮想DOMを意識する必要はありません。これはあくまでReactが内部でコンポーネントを処理し、実際のWebページの表示に反映する際の仕組みなのです。

Reactを使用しても、実際のWebページを構成するDOM（Document Object Model）の更新をページの表示に反映するのは、最終的にはWebブラウザの仕事となります。そしてReactがいくら高速に動作したとしても、DOMを更新するのにはそれなりの時間がかかります。そこでReactでは、本物のDOM以外に自分の管轄下に仮想DOMを構築し、細かな更新はとりあえず仮想DOMに対して実行します。そして仮想DOMの中で更新が必要な差分を割り出します。その後、仮想DOMと本物のDOMとの間でreconciliation（調

停、一致）と呼ばれる処理を実行し、本物のDOMにその差分を反映させる際の変更を最小限にとどめるようにします。それによって高速なページ内容の更新を実現しているのです。このあたりの処理はかなり高度なものですが、Reactを使う際にそれを意識している必要はありません。しかし、そうした優れた仕組みをなんとなくでも知ることで、確信を持ってReactを学び、使いこなすことができるようになるでしょう。