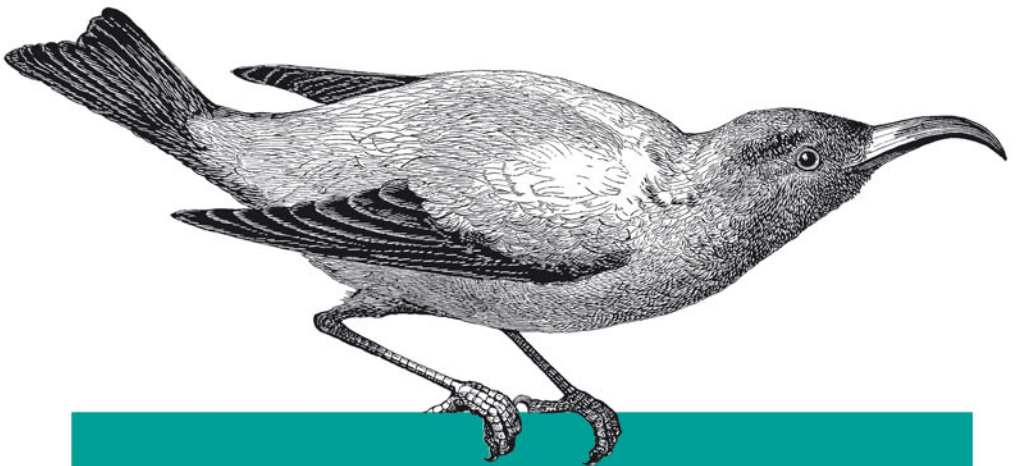
O'REILLY®



React
w działaniu
Tworzenie aplikacji internetowych

Helion

Stoyan Stefanov

Tytuł oryginału: React: Up & Running: Building Web Applications

Tłumaczenie: Joanna Zatorska

ISBN: 978-83-283-3301-7

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/reacwd
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Witaj, świecie

Ruszajmy na wyprawę, podczas której opanujesz tworzenie aplikacji za pomocą technologii React. W tym rozdziale nauczysz się konfigurować Reacta, a także napiszesz swoją pierwszą aplikacją internetową „Witaj, świecie".

Konfiguracja

Najpierw musisz uzyskać kopię biblioteki React. Na szczęście jest to bardzo proste.

Przejdź do strony *http://reactjs.com* (która powinna przekierować Cię na oficjalną stronę GitHub, znajdującą się pod adresem *http://facebook.github.io/react/*) i kliknij przycisk *Download*. Na kolejnej stronie kliknij przycisk *Download Starter Kit*, aby pobrać plik ZIP zawierający bibliotekę. Rozpakuj pobrany plik i skopiuj znajdujący się tam folder do miejsca, które łatwo odszukać.

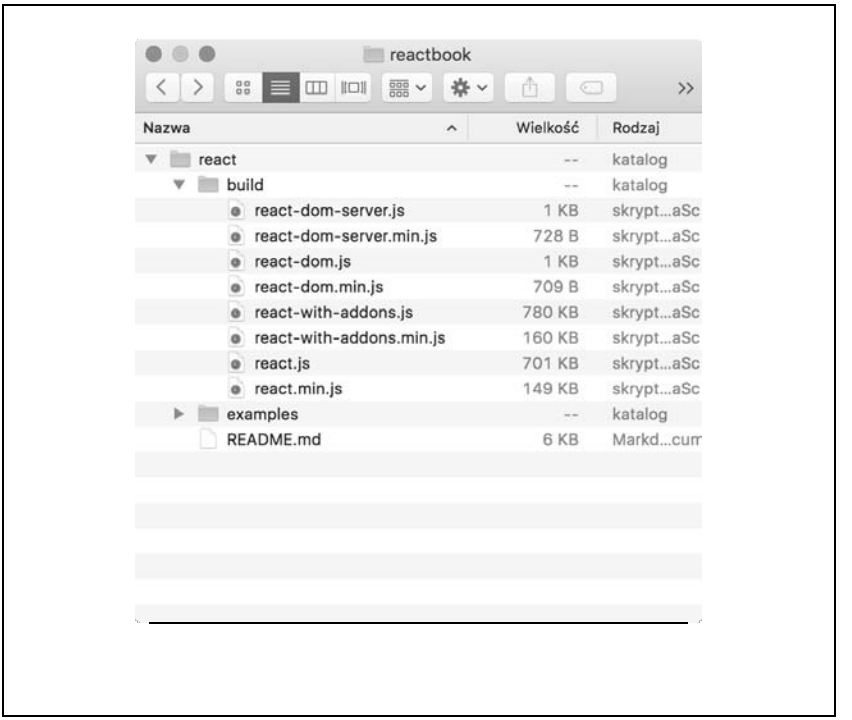
Możesz to zrobić na przykład następująco:

```
mkdir ~/reactbook
mv ~/Downloads/react-0.14.7/ ~/reactbook/react
```

Po wykonaniu tych czynności zawartość katalogu roboczego (o nazwie *reactbook*) powinna wyglądać jak na rysunku 1.1.

Do rozpoczęcia pracy wystarczy plik *~/reactbook/react/build/react.js*. Kolejne pliki poznasz w trakcie dalszej pracy.

Zwróć uwagę na to, że React nie narzuca żadnej struktury folderów. Wspomniany plik możesz umieścić w folderze o dowolnej nazwie, a nazwę samego pliku, *react.js*, możesz dowolnie zmienić.



Rysunek 1.1. Zawartość katalogu z biblioteką React

Witaj, świecie React

Zacznij od utworzenia prostej strony w folderze roboczym (*~/reactbook/01.01.hello.html*):

```
<!DOCTYPE html>
<html>
   <head>
      <title>Witaj, React</title>
      <meta charset="utf-8">
   </head>
   <body>
      <div id="app">
         <!-- tutaj będzie renderowana moja aplikacja -->
      </div>
      <script src="react/build/react.js"></script>
```

```
        <script src="react/build/react-dom.js"></script>
        <script>
            // kod mojej aplikacji
        </script>
    </body>
</html>
```

Kody wszystkich przykładów znajdujących się w książce znajdziesz na stronie *ftp://ftp.helion.pl/przyklady/reacwd.zip*.

W tym pliku warto zwrócić uwagę na następujące aspekty:

- Dołączamy bibliotekę React i jej rozszerzenie DOM (korzystając ze znaczników `<script src>`).
- Definiujemy miejsce, w którym aplikacja powinna być widoczna na stronie (`<div id="app">`).

W aplikacji tworzonej za pomocą Reacta można zawsze korzystać zarówno z kodu HTML, jak i z innych bibliotek JavaScriptu. Na jednej stronie może się znajdować kilka aplikacji napisanych z użyciem Reacta. Wystarczy umieścić odpowiedni kod w strukturze DOM i rozkazać mu, aby „czynił swoją magię".

Dodaj teraz kod wyświetlający powitanie. W tym celu zmodyfikuj plik *01.01.hello.html* i zastąp wiersz `// kod mojej aplikacji` następującym kodem:

```
ReactDOM.render(
    React.DOM.h1(null, "Witaj, świecie!"),
    document.getElementById("app")
);
```

Wczytaj plik *01.01.hello.html* w przeglądarce i sprawdź działanie swojej nowej aplikacji (rysunek 1.2).

Gratulacje! Utworzyłeś swoją pierwszą aplikację za pomocą Reacta!

Na rysunku 1.2 widoczny jest też podgląd *wygenerowanego* kodu w narzędziu Chrome Developer Tools. Możesz zauważyć, że tymczasowa zawartość elementu `<div id="app">` została zastąpiona zawartością wygenerowaną przez aplikację utworzoną za pomocą Reacta.

Rysunek 1.2. Działająca aplikacja Witaj, świecie

Co tu się wydarzyło?

Kod odpowiadający za działanie Twojej pierwszej aplikacji zawiera kilka ciekawych elementów.

Przede wszystkim należy zauważyć użycie obiektu `React`. Za jego pośrednictwem dostępne są wszystkie możliwe do wykorzystania metody API. API jest w rzeczywistości celowo uproszczone i zawiera niewielką liczbę metod, co ułatwia ich zapamiętanie.

Widoczny jest też obiekt `ReactDOM`. W obiekcie tym zdefiniowano tylko kilka metod, z których najbardziej użyteczna jest metoda `render()`. Metody te były wcześniej dostępne z poziomu obiektu `React`, lecz począwszy od wersji 0.14, zostały od niego odseparowane. W ten sposób podkreślono, że samo renderowanie aplikacji jest zupełnie inną kwestią. Można przecież utworzyć aplikację za pomocą Reacta przeznaczoną do renderowania w różnych środowiskach — na przykład w postaci kodu HTML (hierarchii DOM w przeglądarce), w elemencie `canvas` lub w natywnych aplikacjach Android lub iOS.

Kolejnym aspektem jest koncepcja *komponentów*. Interfejs użytkownika tworzy się za pomocą komponentów łączonych ze sobą w sposób, który najlepiej spełnia nasze potrzeby. W swoich aplikacjach będziesz w przyszłości tworzyć własne komponenty, lecz na początku możesz skorzystać z gotowych komponentów Reacta, wykorzystujących elementy DOM języka HTML. Masz do nich dostęp poprzez obiekt `React.DOM`. W pierwszym przykładzie używasz komponentu `h1`. Odpowiada on elementowi HTML `<h1>` i jest dostępny poprzez wywołanie funkcji `React.DOM.h1()`.

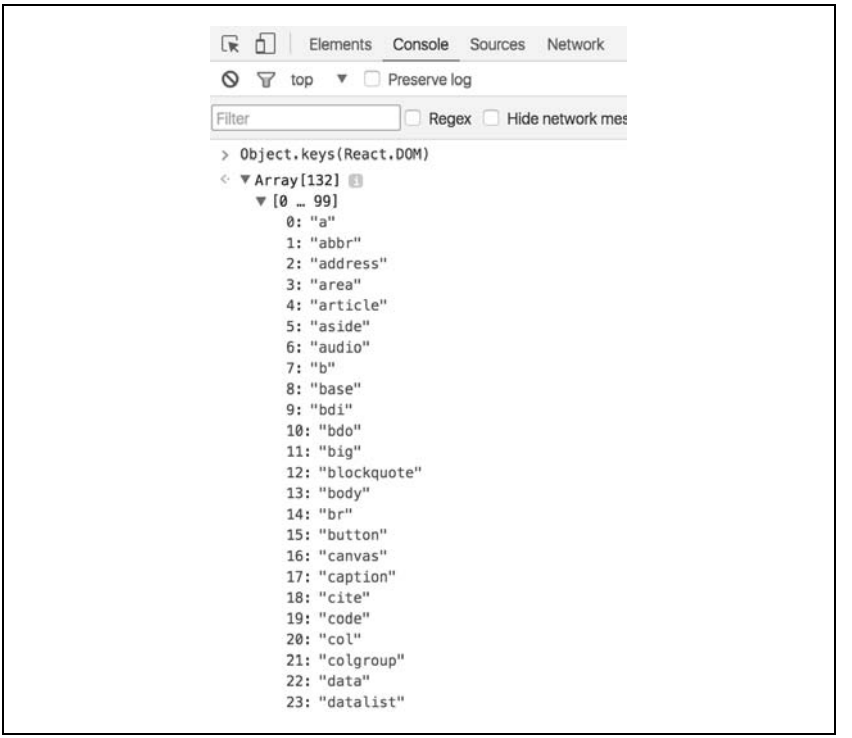
Wreszcie można zauważyć dostęp do elementu DOM za pomocą starej dobrej składni `document.getElementById("app")`. W ten sposób informujemy Reacta, w którym miejscu strony powinna zostać wyświetlona nasza aplikacja. Jest to pomost między znanym już sposobem przetwarzania elementów DOM a nowym sposobem stosowanym w świecie Reacta.

Po przekroczeniu mostu łączącego DOM ze światem Reacta nie musisz się już przejmować przetwarzaniem elementów DOM, ponieważ React przetwarza komponenty w elementy stosowane w docelowej platformie (w przeglądarce DOM, w elemencie `canvas`, w aplikacji natywnej). Nie musisz się przejmować strukturą DOM, ale oczywiście możesz. W bibliotece React znajdują się furtki, które w razie potrzeby umożliwiają powrót do świata DOM.

Wiesz już, za co odpowiada każdy wiersz, i możesz spojrzeć na kod całościowo. Oto co się stało: wyrenderowałeś komponent Reacta w wybranym miejscu drzewa DOM. Zawsze renderowany jest jeden komponent wysokiego poziomu, który może mieć dowolną liczbę komponentów potomnych (które z kolei mogą mieć swoich potomków itd.). Nawet w tym prostym przykładzie komponent `h1` ma potomka — napis „Witaj, świecie!”.

React.DOM.*

Jak już wiesz, poprzez obiekt `React.DOM` możesz skorzystać z wielu elementów HTML reprezentowanych przez komponenty Reacta (rysunek 1.3 przedstawia sposób, który umożliwia wyświetlenie całej listy w konsoli przeglądarki). Przyjrzyjmy się teraz API tego obiektu.



Rysunek 1.3. Lista właściwości React.DOM

Zwróć uwagę na różnice między obiektami `React.DOM` a `ReactDOM`. Pierwszy z nich jest kolekcją gotowych elementów HTML, a drugi służy do renderowania aplikacji w przeglądarce (przypomnij sobie metodę `ReactDOM.render()`).

Przyjrzyjmy się parametrom, jakie przyjmują wszystkie metody `React.DOM.*`. Pamiętaj, że aplikacja „Witaj, świecie!" wygląda następująco:

```
ReactDOM.render(
   React.DOM.h1(null, "Witaj, świecie!"),
   document.getElementById("app")
);
```

Pierwszym parametrem przyjmowanym przez metodę `h1()` (w tym przypadku ma on wartość `null`) jest obiekt określający wszelkie właściwości (pomyśl o atrybutach DOM), które można przekazać do komponentu. Możesz na przykład napisać taki kod:

```
React.DOM.h1(
   {
      id: "my-heading",
   },
   "Witaj, świecie!"
),
```

Na rysunku 1.4 widoczny jest kod HTML wygenerowany w powyższym przykładzie.

Rysunek 1.4. Kod HTML wygenerowany przez wywołanie metody obiektu React.DOM

Drugi parametr (w tym przypadku `"Witaj, świecie!"`) definiuje element potomny komponentu. Najprostszym przypadkiem jest zwykły potomek tekstowy (węzeł `Text` w terminologii DOM), widoczny w poprzednim przykładzie. Można jednak zagnieździć dowolną liczbę elementów potomnych, które należy przekazać jako dodatkowe parametry funkcji. Na przykład:

```
React.DOM.h1(
   {id: "my-heading"},
   React.DOM.span(null, "Witaj,"),
   " świecie!"
),
```

Kolejny przykład zawiera zagnieżdżone komponenty (uzyskany efekt jest przedstawiony na rysunku 1.5):

```
React.DOM.h1(
   {id: "my-heading"},
   React.DOM.span(null,
      React.DOM.em(null, "Wita"),
      "j,"
   ),
   " świecie!"
),
```

Rysunek 1.5. Kod HTML wygenerowany poprzez zagnieżdżone wywołania React.DOM

Jak widać, zagnieżdżanie komponentów może szybko doprowadzić do kodu zawierającego wiele wywołań funkcji i nawiasów, które należy uważnie śledzić. Problem ten można rozwiązać, korzystając ze *składni JSX*. JSX jest techniką wartą omówienia (zajmiemy się nią w rozdziale 4.), lecz na razie pozostańmy przy zwykłej składni JavaScriptu, ponieważ JSX wzbudza pewne kontrowersje. Otóż użytkownicy często na początku reagują na tę składnię dość alergicznie (hmm, XML w skrypcie napisanym w JavaScripcie!), ale później okazuje się ona niezastąpiona. Oto poprzedni przykład, tym razem napisany z użyciem składni JSX:

```
ReactDOM.render(
   <h1 id="my-heading">
      <span><em>Wita</em>j,</span> świecie!
   </h1>,
   document.getElementById("app")
);
```

Specjalne atrybuty DOM

Musisz poznać kilka specjalnych atrybutów DOM, a mianowicie `class`, `for` i `style`.

Atrybutów `class` i `for` nie można używać w JavaScripcie, ponieważ są to słowa zarezerwowane. Zamiast nich używaj elementów `className` i `htmlFor`:

```
// Antyprzykład
// to nie działa
React.DOM.h1(
   {
      class: "pretty",
      for: "me",
   },
   "Witaj, świecie!"
);
// Poprawny przykład
// to działa
React.DOM.h1(
   {
      className: "pretty",
      htmlFor: "me",
   },
   "Witaj, świecie!"
);
```

Przejdźmy teraz do atrybutu `style`, określającego styl. Otóż nie można do niego przekazać zwykłego łańcucha tekstowego, jak w kodzie HTML. W tym przypadku musi to być obiekt JavaScriptu. Zawsze warto unikać stosowania łańcuchów tekstowych, aby zmniejszyć ryzyko ataków typu cross-site scripting (XXS). Z tego względu wspomniana zmiana jest pożądana.

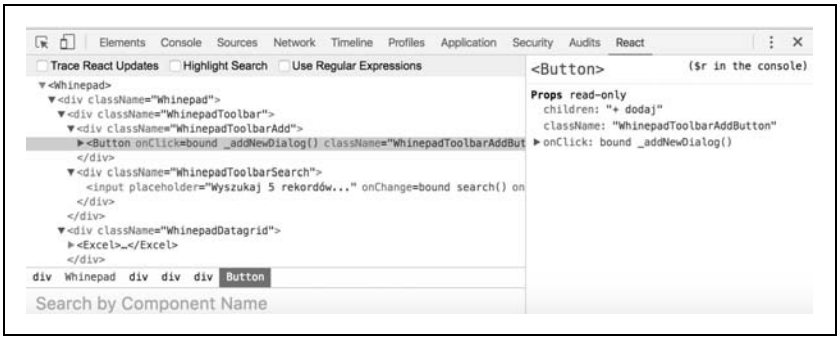
```
// Antyprzykład
// to nie działa
React.DOM.h1(
   {
      style: "background: black; color: white; font-family: Verdana",
   },
   "Witaj, świecie!"
);

// Poprawny przykład
// to działa
React.DOM.h1(
   {
      style: {
         background: "black",
         color: "white",
         fontFamily: "Verdana",
      }
   },
   "Witaj, świecie!"
);
```

Zauważ też, że przy określaniu właściwości CSS musisz użyć nazw API JavaScriptu. Innymi słowy: użyj właściwości `fontFamily` zamiast `font-family`.

Rozszerzenia przeglądarki React DevTools

Jeśli podczas przeglądania niektórych przykładów z tego rozdziału otworzyłeś konsolę przeglądarki, zapewne ujrzałeś informację o treści: „Download the React DevTools and use an HTTP server (instead of a file: URL) for a better development experience: *https://fb.me/react-devtools*". Na stronie o podanym adresie dostępne są łącza umożliwiające zainstalowanie rozszerzenia przeglądarki, które może się okazać przydatne w debugowaniu aplikacji utworzonej z użyciem Reacta (rysunek 1.6).

Na początku rozszerzenie to może się wydawać przytłaczające, lecz wrażenie to minie, zanim dotrzesz do rozdziału 4.



Rysunek 1.6. Rozszerzenie React DevTools

Co dalej: niestandardowe komponenty

Na tym zakończyliśmy pracę nad prostą aplikacją „Witaj, świecie". Wiesz już jak:

- zainstalować, skonfigurować i stosować bibliotekę React (jest to kwestia umieszczenia dwóch znaczników `<script>`);
- wyrenderować komponent Reacta w wybranej lokalizacji DOM (na przykład `ReactDOM.render(reactCo, domGdzie)`);
- używać wbudowanych komponentów, które wykorzystują zwykłe elementy DOM (na przykład `React.DOM.div(atrybuty, dzieci)`).

Jednak prawdziwa potęga Reacta objawia się podczas używania niestandardowych komponentów budujących (i uaktualniających!) interfejs aplikacji. Z kolejnego rozdziału dowiesz się, jak je wykorzystać.

Skorowidz

E

F

H

I

J

K

L

M

N

O

P

R

S

T

U

W

Z

O'REILLY®

React — programowanie może być przyjemne!

Interfejs użytkownika musi działać w przewidywalny sposób na różnych przeglądarkach i urządzeniach. Powinien zapewniać adekwatną obsługę wprowadzanych danych oraz umożliwiać płynne i niezawodne komunikowanie się aplikacji z użytkownikiem. Dziś programiści mogą skupić się na samym działaniu aplikacji, gdyż interfejs użytkownika można szybko, łatwo i przyjemnie zbudować za pomocą biblioteki React.

Poznasz ją dzięki temu przewodnikowi i zaczniesz wykorzystywać w tworzeniu aplikacji internetowych. Dowiesz się, jak można jednorazowo zdefiniować elastyczny interfejs, nauczysz się tworzyć rozbudowane aplikacje z wykorzystaniem małych, łatwych w zarządzaniu komponentów. Zapoznasz się również z JSX — technologią świetnie uzupełniającą React. Nauczysz się też korzystać z dodatkowych narzędzi, takich jak Browserify, Jest, ESLint, Flow, Flux czy Immutable.js. Wzbogać swój warsztat programisty aplikacji WWW!

Stoyan Stefanov — jest inżynierem w Facebooku. Wcześniej pracował dla Yahoo!, gdzie stworzył smush.it — narzędzie online służące do optymalizacji obrazów. Brał też udział w tworzeniu narzędzia o nazwie YSlow 2.0, przeznaczonego do pomiaru wydajności aplikacji. Jest autorem i współautorem wielu książek dotyczących programowania i tworzenia aplikacji internetowych. Prowadzi bloga dostępnego pod adresem *http://phpied.com*. Często zabiera głos na prestiżowych konferencjach, takich jak Velocity, JSConf, Fronteers i wielu innych.

Niektóre zagadnienia omówione w książce:

- przygotowywanie biblioteki React do pracy
- komponenty, ich właściwości, stan i cykl życia
- wykorzystywanie komponentów do budowy UI
- korzystanie z narzędzi pomocniczych przy budowie aplikacji
- diagnostyka i testowanie kodu aplikacji

Helion
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900
Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl
Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci
Informatyka w najlepszym wydaniu
ISBN 978-83-283-3301-7
9 788328 333017
cena: 39,90 zł