REACT I REDUX

Praktyczne tworzenie aplikacji WWW

Tytuł oryginału: Learning React: A Hands-On Guide to Building Web Applications Using React and Redux (2nd Edition)

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-283-4726-7

Authorized translatlon from the English language edition, entitIed: LEARNING REACT: A HANDS-ON GUIDE TO BUILDING WEB APPLICATIONS USING REACT AND REDUX, Second Edition; ISBN 013484355X; by Kirupa Chinnathambi; published by Pearson Education, Inc, publishing as Addison-Wesley Professional.

Helion SA
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/rerew2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

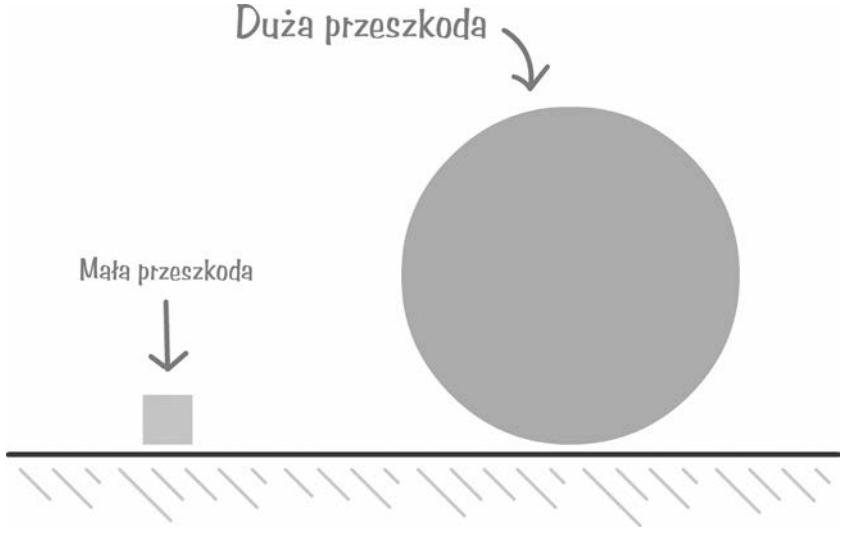
Printed in Poland.

Spis treści

2

Twoja pierwsza aplikacja React

Po przeczytaniu poprzedniego rozdziału prawdopodobnie znasz już całą historię biblioteki React i wiesz, że pomaga ona tworzyć najbardziej skomplikowane interfejsy użytkownika. Jednak z powodu mnogości niezwykłych funkcjonalności, jakie ta biblioteka oferuje, rozpoczęcie korzystania z niej nie jest proste. Droga do poznania tej biblioteki jest stroma, pełna mniejszych i większych przeszkód, co pokazuje rysunek 2.1.



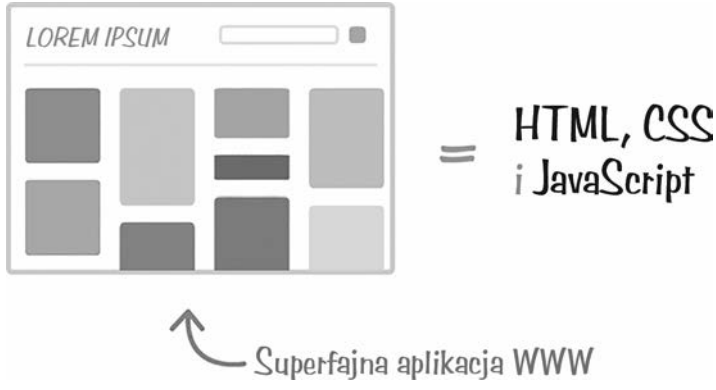
Rysunek 2.1. Przeszkody są różne, jedne małe, inne duże

W tym rozdziale zaczniemy od podstaw i utrudzimy się trochę, tworząc pierwszą prostą aplikację opartą na bibliotece React. Napotkasz przeszkody, z których kilku na razie nie będziesz usuwał. W tym rozdziale nie tylko zbudujesz coś, co z dumą będziesz mógł pokazać rodzinie i przyjaciołom, lecz także porządnie przygotujesz się do następnych rozdziałów, w których będziesz zgłębiał wszystko to, co biblioteka React ma do zaoferowania.

Język JSX

Zanim zaczniesz tworzyć swoją pierwszą aplikację, musisz dowiedzieć się o jednej ważnej rzeczy. Biblioteka React jest niepodobna do innych bibliotek utworzonych w JavaScript, których prawdopodobnie wcześniej używałeś. Zawiedziesz się, jeżeli będziesz ją wprost odnosił do kodu, który napisałeś wcześniej, wykorzystując znacznik `<script>`. Biblioteka React jest pod tym względem irytująco wyjątkowa i zmienia sposób tworzenia aplikacji.

Jak wiesz, aplikacja WWW (i wszystko inne, co wyświetla przeglądarka) składa się z kodów HTML, CSS i JavaScript (patrz rysunek 2.2).



Rysunek 2.2. Aplikacja WWW składa się z kodów HTML, CSS i JavaScript

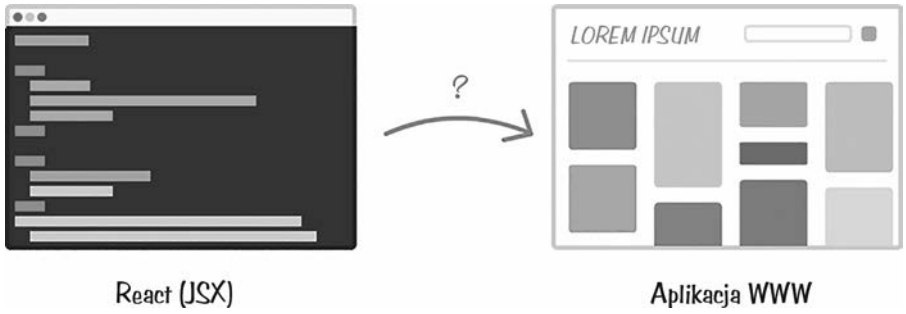
Nie ma znaczenia, czy aplikacja została napisana z użyciem biblioteki React czy innej, na przykład Angular, Knockout lub jQuery. **Końcowy produkt** musi być kombinacją kodów HTML, CSS i JavaScript. W przeciwnym wypadku przeglądarka nie będzie „wiedzieć", co ma robić.

W tym momencie ujawnia się wyjątkowa natura biblioteki React. W kodzie aplikacji, owszem, wykorzystuje się języki HTML, CSS i JavaScript, ale głównie stosuje się język JSX. Jak wspomniałem w rozdziale 1. „Wstęp do biblioteki React", JSX jest językiem, w którym łatwo definiuje się elementy i funkcjonalności interfejsu użytkownika, łącząc kod JavaScript ze znacznikami HTML. Brzmi to ciekawie (język JSX zobaczysz w akcji już za chwilę), ale pojawia się pewien mały problem. Twoja przeglądarka nie ma pojęcia, co ma robić z kodem JSX.

Aby utworzyć aplikację przy użyciu biblioteki React, trzeba znaleźć sposób na przekształcenie kodu JSX na stary, dobry, zrozumiały dla przeglądarki kod JavaScript (patrz rysunek 2.3).

Jeżeli się tego nie zrobi, aplikacja React po prostu nie będzie działać. Nie brzmi to dobrze. Na szczęście istnieją dwa rozwiązania tego problemu:

1. **Utworzenie środowiska programistycznego opartego na platformie Node i kilku dodatkowych narzędziach.** W takim przypadku przy każdej kompilacji kod JSX będzie automatycznie przekształcany w kod JavaScript i zapisywany w pliku podobnym do wielu innych plików tego rodzaju.
2. **Automatyczne przekształcanie kodu JSX w JavaScript przez przeglądarkę w trakcie działania aplikacji.** W takim przypadku kod aplikacji pisze się zwyczajnie w języku JSX, tak jak w JavaScript, a całą resztą zajmuje się przeglądarka.



Rysunek 2.3. Kod JSX trzeba zamienić w coś, co przeglądarka rozumie

Oba rozwiązania mają swoje miejsce w naszym świecie, musisz jednak poznać skutki stosowania każdego z nich.

Pierwsze rozwiązanie, na pozór dość skomplikowane i czasochłonne, jest obecnie stosowane przy tworzeniu nowoczesnych aplikacji WWW. Poza tym, że wymaga ono kompilowania (a ściślej: **transpilowania**) kodu JSX na JavaScript, pozwala także wykorzystywać różne moduły, narzędzia programistyczne i różnorakie funkcjonalności, dzięki którym tworzenie skomplikowanych aplikacji WWW jest prostsze.

W drugim rozwiązaniu droga do celu jest szybsza i prostsza. Wprawdzie więcej czasu trzeba poświęcić na pisanie kodu, ale za to mniej na majstrowanie przy środowisku programistycznym. Aby wykorzystać ten sposób, wystarczy w kodzie aplikacji umieścić odwołanie do pewnego pliku ze skryptem. Skrypt ten zamienia kod JSX na JavaScript podczas ładowania strony. Dzięki temu Twoja aplikacja zacznie żyć, mimo że nie musiałeś nic specjalnego robić ze środowiskiem programistycznym.

W swojej pierwszej przygodzie z biblioteką React wykorzystasz drugie rozwiązanie. Zapewne dziwisz się, że nie można zawsze z niego korzystać. Powód jest taki, że Twoja przeglądarka przy każdym użyciu aplikacji będzie dodatkowo obciążana przekształcaniem kodu JSX w JavaScript. Jest to rozwiązanie całkowicie do przyjęcia w trakcie nauki, ale niedopuszczalne w rzeczywistych aplikacjach. Dlatego później, gdy już dobrze zaznajomisz się z biblioteką React, przejrzymy jeszcze raz cały przerobiony materiał i przygotujemy środowisko programistyczne.

Pierwsze kroki z React

W poprzedniej części rozdziału poznałeś dwa sposoby uczynienia aplikacji React zrozumiałą dla przeglądarki. W tej części przekujemy teorię na praktykę. Na początku będzie potrzebna pusta strona HTML.

Utwórz nowy dokument HTML zawierający następujący treść:

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>React! React! React!</title>
```

```
</head>

<body>
  <script>

  </script>
</body>

</html>
```

Ta strona nie ma w sobie nic ciekawego, więc dodajmy do niej odwołanie do biblioteki React. Zaraz pod znacznikiem `<title>` wpisz następujące wiersze:

```
<script src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
```

Powyższy kod ładuje podstawową bibliotekę React oraz różne dodatkowe rzeczy niezbędne do korzystania z modelu DOM. Bez tych wierszy w ogóle nie da się utworzyć aplikacji opartej na bibliotece React.

Ale to nie wszystko. Trzeba dodać odwołanie do jeszcze jednej biblioteki. Tuż pod powyższymi znacznikami `<script>` wpisz następujący wiersz:

```
<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
```

Jest to odwołanie do kompilatora Babel (*http://babeljs.io*). Kompilator ten robi wiele fajnych rzeczy, z których najważniejszą dla nas jest przekształcanie kodu JSX na JavaScript.

W tym momencie Twoja strona HTML powinna wyglądać jak niżej:

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>React! React! React!</title>
  <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
  <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
</head>

<body>
  <script>

  </script>
</body>

</html>
```

Jeżeli otworzysz teraz tę stronę w przeglądarce stwierdzisz, że jest pusta. To prawidłowy efekt. Zaraz się tym zajmiemy.

Wyświetlenie imienia

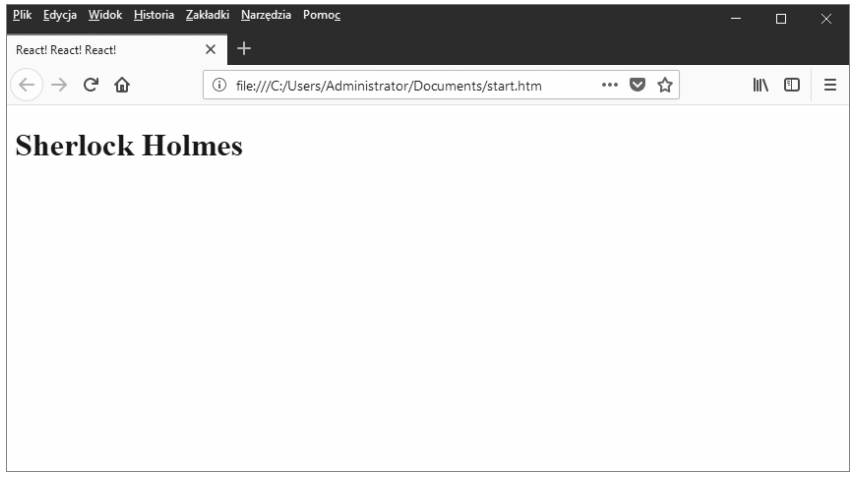
Teraz za pomocą biblioteki React umieścisz na stronie swoje imię. W tym celu użyjesz metody `render()`. Wewnątrz pustego znacznika `<script>` wpisz następujący kod:

```
ReactDOM.render(
  <h1>Sherlock Holmes</h1>,
  document.body
);
```

Nie przejmuj się, jeżeli coś wydaje Ci się bez sensu. Naszym celem jest wyświetlenie na stronie czegokolwiek. Sens temu nadamy później. Teraz, zanim otworzysz stronę i sprawdzisz, co się dzieje, musisz oznakować swój skrypt, aby kompilator mógł użyć całej swojej magii. Umieść w znaczniku `<script>` atrybut `type` z wartością `text/babel`:

```
<script type="text/babel">
  ReactDOM.render(
    <h1>Sherlock Holmes</h1>,
    document.body
  );
</script>
```

Po wprowadzeniu powyższych zmian otwórz stronę w przeglądarce. Zobaczysz wypisane ogromnymi literami słowa `Sherlock Holmes`, jak na rysunku 2.4.



Rysunek 2.4. Twoja przeglądarka powinna wyświetlić słowa „Sherlock Holmes”

Gratulacje! Właśnie utworzyłeś przy użyciu biblioteki React swoją pierwszą aplikację.

W tej aplikacji nie ma się czym ekscytować. Zacznijmy od tego, że raczej nie nazywasz się Sherlock Holmes. Aplikacja wprawdzie niewiele robi, ale stanowi wprowadzenie do jednej z najczęściej stosowanych metod w świecie React: `ReactDOM.render()`.

Metoda `render()` ma dwa argumenty:

1. kod podobny do HTML (czyli JSX), który ma być wyświetlony
2. miejsce w modelu DOM, w którym biblioteka React ma wyświetlić kod

Nasza metoda `render()` wygląda tak:

```
ReactDOM.render(
  <h1>Sherlock Holmes</h1>,
  document.body
);
```

Pierwszym argumentem metody jest tekst `Sherlock Holmes` umieszczony wewnątrz znaczników `<h1>`. Ten osadzony w kodzie JavaScript kod podobny do HTML to jest właśnie język JSX. W dalszej części książki poświęcimy mnóstwo czasu na zgłębianie tego języka, ale jedną rzecz muszę Ci oznajmić już teraz: *właśnie tak przedziwnie wygląda ten kod*. Zawsze, gdy w kodzie JavaScript widzę nawiasy i ukośniki, coś we mnie w środku zamiera, bo jazgot cudzysłowów i znaków wyjścia zagłusza to, co chcę wyrazić. W języku JSX jest inaczej. Wpisuje się po prostu kod HTML tak jak wyżej. W jakiś magiczny sposób to działa.

Drugi argument metody to `document.body`. Nic w nim nadzwyczajnego nie ma. Argument ten po prostu określa miejsce w modelu DOM, w którym zostaną umieszczone znaczniki powstałe z przekształcenia kodu JSX. W tym przykładzie znacznik `<h1>` (i wszystko wewnątrz niego) zostanie umieszczony w elemencie `body` dokumentu.

Wróćmy do pierwotnego celu, którym było wyświetlenie na stronie nie dowolnego, ale **Twojego** imienia. Zmień zatem odpowiednio swój kod. W moim przypadku metoda `render()` wygląda następująco:

```
ReactDOM.render(
  <h1>Batman</h1>,
  document.body
);
```

Wyszło na to, że mam na imię Batman! Tak czy owak, jeżeli teraz otworzysz stronę, zobaczysz zamiast `Sherlock Holmes` swoje imię.

To wszystko jest dobrze znane

Dzięki językowi JSX kod JavaScript wygląda jak nowy, a końcowy produkt trafiający do przeglądarki jest czystą i schludną kombinacją kodów HTML, CSS i JavaScript. Aby się o tym przekonać, wprowadźmy w wyglądzie i działaniu aplikacji kilka zmian.

Zmiana miejsca docelowego

Najpierw zmienimy miejsce, w którym będzie umieszczony wynik kodu JSX. Umieszczanie treści bezpośrednio w elemencie `<body>` jest zdecydowanie złą praktyką. Wiele rzeczy może pójść nie tak jak trzeba, szczególnie gdy miesza się bibliotekę React z innymi bibliotekami i platformami. Zaleca się więc tworzenie osobnego elementu pełniącego funkcję nowego elementu głównego. Staje się on nowym miejscem docelowym wykorzystywanym przez metodę `render()`. Aby zastosować się do tych zaleceń, wróć do kodu HTML i umieścić w nim element `<div>` z atrybutem `id` o wartości `container`, jak niżej:

```
<body>
  <div id="container"></div>
  <script type="text/babel">
    ReactDOM.render(
      <h1>Batman</h1>,
      document.body
    );
  </script>
</body>
```

Po zdefiniowaniu bezpiecznego elementu `<div>` zmodyfikujmy metodę `render()` tak, aby korzystała z tego elementu zamiast z `document.body`. Poniżej przedstawiony jest jeden ze sposobów, jak to zrobić:

```
ReactDOM.render(
  <h1>Batman</h1>,
  document.querySelector("#container")
);
```

Innym rozwiązaniem jest dodanie nieco kodu poza samą metodą `render()`:

```
var destination = document.querySelector("#container");

ReactDOM.render(
  <h1>Batman</h1>,
  destination
);
```

Zwróć uwagę, że zmienna `destination` zawiera referencję do elementu `container` w modelu DOM. Dzięki temu wewnątrz metody `render()` zamiast pełnej nazwy elementu wystarczy użyć zmiennej `destination`. Powód wykonania tej operacji jest prosty: chciałem w ten sposób pokazać Ci, że cały czas używasz języka JavaScript, a `render()` jest następną nudną metodą z dwoma argumentami.

Trochę stylu!

Wprowadźmy jeszcze jedną zmianę, zanim skończymy na dziś. W tej chwili przeglądarka wyświetla Twoje imię, wykorzystując domyślny styl nagłówka `<h1>`. Wygląda on okropnie, więc zmieńmy go, dodając nieco kodu CSS. Wewnątrz znacznika `<head>` wstaw blok `<style>` z następującą zawartością:

```
<style>
  #container {
    padding: 50px;
    background-color: #EEE;
  }
  #container h1 {
    font-size: 144px;
    font-family: sans-serif;
    color: #0080A8;
  }
</style>
```

Po wprowadzeniu zmian otwórz stronę w przeglądarce. Zwróć uwagę, że teraz tekst wygląda nieco okazalej niż wcześniej, gdy zastosowany był domyślny styl nagłówka `<h1>` (patrz rysunek 2.5).

To działa, ponieważ kiedy został wykonany kod biblioteki React, w elemencie `<body>` znalazł się element `container` ze znacznikiem `<h1>` w środku. Nie ma znaczenia, że znacznik ten został w pełni zdefiniowany za pomocą języków JavaScript i JSX ani że style CSS znajdują się zupełnie poza metodą `render()`. Końcowa aplikacja React składa się w 100% z czystej kombinacji kodów HTML, CSS i JavaScript. Wynik transpilacji wygląda mniej więcej tak:

Rysunek 2.5. Efekt dodania stylu CSS

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>React! React! React!</title>

  <style>
    #container {
      padding: 50px;
      background-color: #EEE;
    }
    #container h1 {
      font-size: 144px;
      font-family: sans-serif;
      color: #0080A8;
    }
  </style>
</head>

<body>
  <div id="container">
    <h1>Batman</h1>
  </div>
</body>

</html>
```

Zauważ, że w powyższym kodzie nie ma ani śladu po bibliotece React.

Podsumowanie

Jeżeli była to Twoja pierwsza aplikacja React, dowiedziałeś się o niej mnóstwa podstawowych rzeczy. Jedną z najważniejszych jest to, że React różni się od innych bibliotek, ponieważ do definiowania elementów interfejsu użytkownika wykorzystuje się w niej całkowicie nowy język JSX. Poznałeś już go nieco, gdy definiowałeś znacznik `<h1>` wewnątrz metody `render()`.

Możliwości języka JSX wykraczają daleko poza definiowanie elementów interfejsu użytkownika. Przede wszystkim całkowicie zmienia on sposób tworzenia aplikacji. Ponieważ przeglądarka „nie rozumie" kodu JSX w jego naturalnej postaci, trzeba wykonywać pośredni krok i przekształcać go na kod JavaScript. Jedno z podejść polega na transpilowaniu kodu JSX na odpowiadający mu kod JavaScript. Innym rozwiązaniem (wykorzystanym w tym rozdziale) jest zastosowanie kompilatora Babel, który przekształca kod JSX na JavaScript w samej przeglądarce. Ponieważ to podejście ma negatywny wpływ na wydajność przeglądarki, jego stosowanie nie jest zalecane w rzeczywistych, produkcyjnych aplikacjach. Jednak w trakcie poznawania biblioteki React możesz sobie pozwolić na ten luksus.

W następnych rozdziałach poświęcimy nieco czasu na zgłębianie języka JSX i innych metod, które obok `render()` stanowią o zaletach biblioteki React.

> **Jeżeli napotkasz problemy, pytaj!**
>
> Jeżeli będziesz miał jakiekolwiek pytania albo Twój kod nie będzie działał zgodnie z oczekiwaniami, pytaj śmiało! Wejdź na forum *https://forum.kirupa.com* i korzystaj z pomocy najsympatyczniejszych i najbardziej kompetentnych ludzi w internecie!

OTO REACT:
ZNAKOMITY EFEKT W KRÓTKIM CZASIE!

React służy do budowy złożonych jednostronicowych aplikacji WWW. Jest biblioteką języka JavaScript, utworzoną i udostępnianą przez Facebook na licencji open source. Biblioteka ta oferuje wiele gotowych komponentów i innych przydatnych funkcji. Pozwala rozwiązywać często powtarzające się i uciążliwe problemy programistyczne w zaskakująco prosty sposób. Pierwsze próby programowania przy użyciu biblioteki React mogą jednak sprawiać trudności. Podobnie jak z innymi narzędziami dla profesjonalistów — aby docenić jej zalety, trzeba ją poznać.

Ta książka jest jedynym w swoim rodzaju praktycznym przewodnikiem po bibliotece React — przejrzystym i przystępnym. Zawiera wskazówki ułatwiające błyskawiczny start w tworzeniu efektownych i efektywnych aplikacji WWW. Nawet programista, który pierwszy raz ma do czynienia z tym narzędziem, będzie mógł w krótkim czasie napisać i uruchomić swoją aplikację. W książce zamieszczono setki przykładów omawiających krok po kroku zastosowanie poszczególnych funkcji, a złożone pojęcia wyjaśniono za pomocą trafnych ilustracji. W ten sposób można sobie znacznie uprościć tworzenie nawet bardzo skomplikowanych elementów interfejsu aplikacji!

W tej książce między innymi:

- tworzenie aplikacji za pomocą biblioteki React
- pisanie komponentów definiujących elementy interfejsu użytkownika
- zarządzanie stanami aplikacji i jej danymi
- cykl życia komponentów
- tworzenie wielostronicowych aplikacji

KIRUPA CHINNATHAMBI — pierwsze linie kodu napisał na początku lat 90. Nieco później, zanim jeszcze pojawiło się słowo „blog", zaczął umieszczać porady dla programistów na stronie *kirupa.com*. W kolejnych latach napisał setki artykułów i kilka książek. Wkrótce po ukończeniu MIT zaczął pracę w firmie Microsoft, jako program manager — w ramach projektu znanego dziś jako Visual Studio.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-4726-7

9 788328 347267

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 59,00 zł

Pearson
Addison-Wesley