

Spis treści

Wstęp	11
Przedmowa	13
Podziękowania	15
O książce	17

CZĘŚĆ I. PODSTAWY NODE 19

Rozdział 1. Witamy w Node.js 21

1.1.	Node jest zbudowane w oparciu o JavaScript	22
1.2.	Asynchroniczna i oparta na zdarzeniach: przeglądarka internetowa	23
1.3.	Asynchroniczny i oparty na zdarzeniach: serwer	25
1.4.	Aplikacje DIRT	27
1.5.	Domyślna aplikacja jest typu DIRT	29
1.5.1.	Prosty przykład aplikacji asynchronicznej	30
1.5.2.	Serwer HTTP	30
1.5.3.	Strumieniowanie danych	32
1.6.	Podsumowanie	33

Rozdział 2. Tworzenie aplikacji wielopokojowego czatu 35

2.1.	Ogólny opis aplikacji	36
2.2.	Wymagania aplikacji i konfiguracja początkowa	38
2.2.1.	Obsługa HTTP i WebSocket	39
2.2.2.	Tworzenie struktury plików aplikacji	39
2.2.3.	Wskazanie zależności	40
2.2.4.	Instalacja zależności	41
2.3.	Udostępnianie plików HTML, CSS i kodu JavaScript działającego po stronie klienta	41
2.3.1.	Tworzenie podstawowego serwera plików statycznych	42
2.3.2.	Dodanie plików HTML i CSS	45
2.4.	Obsługa wiadomości czatu za pomocą biblioteki Socket.IO	46
2.4.1.	Konfiguracja serwera Socket.IO	48
2.4.2.	Obsługa zdarzeń oraz scenariuszy w aplikacji	49
2.5.	Użycie kodu JavaScript działającego po stronie klienta do utworzenia interfejsu użytkownika aplikacji	54
2.5.1.	Przekazywanie do serwera wiadomości oraz żądań zmiany pokoju lub nazwy użytkownika	54
2.5.2.	Wyświetlenie w interfejsie użytkownika wiadomości i listy dostępnych pokoi	55
2.6.	Podsumowanie	58

Rozdział 3. Podstawy programowania w Node 61

- 3.1. Organizacja i wielokrotne użycie kodu Node 62
 - 3.1.1. Tworzenie modułu 64
 - 3.1.2. Dostrajanie tworzenia modułu za pomocą `module.exports` 66
 - 3.1.3. Wielokrotne użycie modułów za pomocą katalogu `node_modules` 68
 - 3.1.4. Zastrzeżenia 68
- 3.2. Techniki programowania asynchronicznego 69
 - 3.2.1. Użycie wywołań zwrotnych do obsługi zdarzeń jednorazowych 71
 - 3.2.2. Użycie emitera zdarzeń do obsługi powtarzających się zdarzeń 74
 - 3.2.3. Wyzwania pojawiające się podczas programowania asynchronicznego 82
- 3.3. Sekwencja logiki asynchronicznej 84
 - 3.3.1. Kiedy stosować szeregową kontrolę przepływu? 85
 - 3.3.2. Implementacja szeregowej kontroli przepływu 86
 - 3.3.3. Implementacja równoległej kontroli przepływu 89
 - 3.3.4. Użycie narzędzi opracowanych przez społeczność 91
- 3.4. Podsumowanie 92

CZĘŚĆ II. TWORZENIE APLIKACJI SIECIOWYCH W NODE 95

Rozdział 4. Tworzenie aplikacji sieciowej w Node 97

- 4.1. Podstawy dotyczące serwera HTTP 99
 - 4.1.1. Jak przychodzące żądania HTTP są przez Node przedstawiane programiście? 99
 - 4.1.2. Prosty serwer HTTP odpowiadający komunikatem „Witaj, świecie” 100
 - 4.1.3. Odczyt nagłówków żądania i zdefiniowanie nagłówków odpowiedzi 101
 - 4.1.4. Ustawienie kodu stanu odpowiedzi HTTP 102
- 4.2. Tworzenie usługi sieciowej RESTful 102
 - 4.2.1. Tworzenie zasobów za pomocą żądań POST 103
 - 4.2.2. Pobieranie zasobów za pomocą żądania GET 105
 - 4.2.3. Usunięcie zasobu za pomocą żądania DELETE 107
- 4.3. Udostępnianie plików statycznych 108
 - 4.3.1. Tworzenie serwera plików statycznych 109
 - 4.3.2. Obsługa błędów serwera 112
 - 4.3.3. Wyprzedzająca obsługa błędów za pomocą wywołania `fs.stat()` 113
- 4.4. Akceptacja danych wejściowych użytkownika przekazanych za pomocą formularza sieciowego 114
 - 4.4.1. Obsługa wysłanych pól formularza sieciowego 114
 - 4.4.2. Obsługa przekazanych plików za pomocą `formidable` 118
 - 4.4.3. Sprawdzanie postępu operacji przekazywania plików 122
- 4.5. Zabezpieczanie aplikacji dzięki użyciu protokołu HTTPS 122
- 4.6. Podsumowanie 124

Rozdział 5. Przechowywanie danych aplikacji Node 125

- 5.1. Niewymagający serwera magazyn danych 126
 - 5.1.1. Magazyn danych w pamięci 126
 - 5.1.2. Magazyn danych oparty na plikach 127

- 5.2. System zarządzania relacyjną bazą danych 130
 - 5.2.1. *MySQL* 131
 - 5.2.2. *PostgreSQL* 139
- 5.3. Bazy danych typu NoSQL 141
 - 5.3.1. *Redis* 141
 - 5.3.2. *MongoDB* 146
 - 5.3.3. *Mongoose* 149
- 5.4. Podsumowanie 151

Rozdział 6. *Framework Connect* 153

- 6.1. Konfiguracja aplikacji *Connect* 154
- 6.2. Jak działa metoda pośrednicząca frameworka *Connect*? 155
 - 6.2.1. *Metody pośredniczące wyświetlające żądanie* 156
 - 6.2.2. *Metoda pośrednicząca udzielająca odpowiedzi w postaci komunikatu „Witaj, świecie”* 157
- 6.3. Dlaczego kolejność metod pośredniczących ma znaczenie? 158
 - 6.3.1. *Kiedy metoda pośrednicząca nie wywołuje next()?* 158
 - 6.3.2. *Użycie kolejności metod pośredniczących do przeprowadzenia uwierzytelnienia* 159
- 6.4. Montowanie metody pośredniczącej i serwera 160
 - 6.4.1. *Metody pośredniczące przeprowadzające uwierzytelnianie* 161
 - 6.4.2. *Metoda pośrednicząca wyświetlająca panel administracyjny* 162
- 6.5. Tworzenie konfigurowalnej metody pośredniczącej 164
 - 6.5.1. *Tworzenie konfigurowalnej metody pośredniczącej logger()* 164
 - 6.5.2. *Tworzenie metody pośredniczącej router()* 166
 - 6.5.3. *Tworzenie metody pośredniczącej przeznaczonej do przepisywania adresów URL* 168
- 6.6. Użycie metody pośredniczącej do obsługi błędów 170
 - 6.6.1. *Domyślna obsługa błędów w Connect* 170
 - 6.6.2. *Samodzielna obsługa błędów aplikacji* 171
 - 6.6.3. *Użycie wielu metod pośredniczących przeznaczonych do obsługi błędów* 172
- 6.7. Podsumowanie 176

Rozdział 7. *Metody pośredniczące frameworka Connect* 177

- 7.1. Metody pośredniczące przeznaczone do przetwarzania plików cookie, danych żądań i ciągów tekstowych zapytań 179
 - 7.1.1. *cookieParser()* — przetwarzanie plików cookie 179
 - 7.1.2. *bodyParser()* — przetwarzanie danych żądania 182
 - 7.1.3. *limit()* — ograniczenie danych żądania 184
 - 7.1.4. *query()* — analizator ciągu tekstowego zapytania 186
- 7.2. Metody pośredniczące implementujące podstawowe funkcje wymagane przez aplikację sieciową 187
 - 7.2.1. *logger()* — rejestracja informacji o żądaniu 188
 - 7.2.2. *favicon()* — obsługa ikon *favicon* 191
 - 7.2.3. *methodOverride()* — nieprawdziwe metody *HTTP* 191
 - 7.2.4. *vhost()* — wirtualny hosting 194
 - 7.2.5. *session()* — zarządzanie sesją 195

- 7.3. Metody pośredniczące zapewniające bezpieczeństwo aplikacji sieciowej 200
 - 7.3.1. *basicAuth()* — uwierzytelnianie podstawowe HTTP 200
 - 7.3.2. *csrf()* — ochrona przed atakami typu CSRF 202
 - 7.3.3. *errorHandler()* — obsługa błędów w trakcie tworzenia aplikacji 203
- 7.4. Metody pośredniczące przeznaczone do udostępniania plików statycznych 205
 - 7.4.1. *static()* — udostępnianie plików statycznych 205
 - 7.4.2. *compress()* — kompresja plików statycznych 207
 - 7.4.3. *directory()* — wyświetlenie katalogów 209
- 7.5. Podsumowanie 210

Rozdział 8. Framework Express 213

- 8.1. Utworzenie szkieletu aplikacji 215
 - 8.1.1. Globalna instalacja frameworka Express 216
 - 8.1.2. Generowanie aplikacji 218
 - 8.1.3. Poznanie aplikacji 218
- 8.2. Konfiguracja frameworka Express i tworzonej aplikacji 220
 - 8.2.1. Konfiguracja na podstawie środowiska 221
- 8.3. Generowanie widoków aplikacji Express 223
 - 8.3.1. Konfiguracja systemu widoków 224
 - 8.3.2. Wyszukiwanie widoku 225
 - 8.3.3. Udostępnianie danych widokom 228
- 8.4. Obsługa formularzy i przekazywania plików 232
 - 8.4.1. Implementacja modelu zdjęcia 233
 - 8.4.2. Tworzenie formularza przeznaczonego do przekazywania zdjęć 233
 - 8.4.3. Wyświetlenie listy przekazanych zdjęć 236
- 8.5. Obsługa pobierania zasobów 237
 - 8.5.1. Tworzenie trasy dla pobierania zdjęć 237
 - 8.5.2. Implementacja trasy pobierania zdjęcia 238
- 8.6. Podsumowanie 240

Rozdział 9. Zaawansowane użycie frameworka Express 241

- 9.1. Uwierzytelnianie użytkowników 242
 - 9.1.1. Pisywanie i wczytywanie użytkowników 243
 - 9.1.2. Rejestrowanie nowego użytkownika 248
 - 9.1.3. Logowanie zarejestrowanych użytkowników 254
 - 9.1.4. Metoda pośrednicząca przeznaczona do wczytywania użytkownika 257
- 9.2. Zaawansowane techniki routingu 259
 - 9.2.1. Weryfikacja użytkownika podczas przesyłania treści 260
 - 9.2.2. Metoda pośrednicząca charakterystyczna dla trasy 263
 - 9.2.3. Implementacja stronicowania 266
- 9.3. Utworzenie publicznego API REST 270
 - 9.3.1. Projekt API 270
 - 9.3.2. Dodanie uwierzytelnienia podstawowego 271
 - 9.3.3. Implementacja routingu 272
 - 9.3.4. Włączenie negocjacji treści 275
- 9.4. Obsługa błędów 277
 - 9.4.1. Obsługa błędów 404 278
 - 9.4.2. Obsługa błędów 280
- 9.5. Podsumowanie 283

Rozdział 10. Testowanie aplikacji Node 285

- 10.1. Testy jednostkowe 286
 - 10.1.1. Moduł *assert* 287
 - 10.1.2. Framework *nodeunit* 291
 - 10.1.3. *Mocha* 293
 - 10.1.4. Framework *Vows* 298
 - 10.1.5. Biblioteka *should.js* 301
- 10.2. Testy akceptacyjne 303
 - 10.2.1. *Tobi* 303
 - 10.2.2. *Soda* 305
- 10.3. Podsumowanie 307

Rozdział 11. Szablony w aplikacji sieciowej 309

- 11.1. Użycie szablonów w celu zachowania przejrzystości kodu 310
 - 11.1.1. Szablon w akcji 311
- 11.2. Silnik szablonów *Embedded JavaScript* 314
 - 11.2.1. Tworzenie szablonu 315
 - 11.2.2. Praca z danymi szablonu za pomocą filtrów *EJS* 316
 - 11.2.3. Integracja *EJS* w aplikacji 320
 - 11.2.4. Użycie *EJS* w aplikacjach działających po stronie klienta 321
- 11.3. Użycie języka szablonów *Mustache* wraz z silnikiem *Hogan* 322
 - 11.3.1. Tworzenie szablonu 322
 - 11.3.2. Znaczniki *Mustache* 323
 - 11.3.3. Dostosowanie szablonu *Hogan* do własnych potrzeb 325
- 11.4. Szablony *Jade* 326
 - 11.4.1. Podstawy szablonów *Jade* 328
 - 11.4.2. Logika w szablonach *Jade* 330
 - 11.4.3. Organizacja szablonów *Jade* 333
- 11.5. Podsumowanie 337

CZĘŚĆ III. CO DALEJ? 339

Rozdział 12. Wdrażanie aplikacji Node i zapewnienie bezawaryjnego działania 341

- 12.1. Hosting aplikacji Node 342
 - 12.1.1. Serwery dedykowane i VPS 343
 - 12.1.2. Hosting w chmurze 343
- 12.2. Podstawy wdrożenia 346
 - 12.2.1. Wdrożenie z repozytorium *Git* 346
 - 12.2.2. Zapewnienie działania aplikacji Node 347
- 12.3. Maksymalizacja wydajności i czasu bezawaryjnego działania aplikacji 348
 - 12.3.1. Zapewnienie działania aplikacji za pomocą *Upstart* 349
 - 12.3.2. API klastra — wykorzystanie zalety w postaci wielu rdzeni 351
 - 12.3.3. Proxy i hosting plików statycznych 353
- 12.4. Podsumowanie 354

Rozdział 13. Nie tylko serwery WWW 355

- 13.1. Biblioteka Socket.IO 356
 - 13.1.1. Tworzenie minimalnej aplikacji Socket.IO 357
 - 13.1.2. Użycie biblioteki Socket.IO do odświeżenia strony i stylów CSS 359
 - 13.1.3. Inne zastosowania dla biblioteki Socket.IO 362
- 13.2. Dokładniejsze omówienie sieci TCP/IP 363
 - 13.2.1. Praca z buforami i danymi binarnymi 363
 - 13.2.2. Tworzenie serwera TCP 365
 - 13.2.3. Tworzenie klienta TCP 369
- 13.3. Narzędzia przeznaczone do pracy z systemem operacyjnym 371
 - 13.3.1. Obiekt process, czyli globalny wzorzec Singleton 371
 - 13.3.2. Użycie modułu filesystem 375
 - 13.3.3. Tworzenie procesów zewnętrznych 379
- 13.4. Tworzenie narzędzi powłoki 384
 - 13.4.1. Przetwarzanie argumentów podanych w powłoce 385
 - 13.4.2. Praca ze standardowym wejściem i wyjściem 386
 - 13.4.3. Dodanie koloru do danych wyjściowych 388
- 13.5. Podsumowanie 391

Rozdział 14. Ekosystem Node 393

- 14.1. Dostępne w internecie zasoby dla programistów Node 394
 - 14.1.1. Node i odniesienia do modułów 394
 - 14.1.2. Grupy Google 395
 - 14.1.3. IRC 396
 - 14.1.4. Zgłaszanie problemów w serwisie GitHub 397
- 14.2. Serwis GitHub 398
 - 14.2.1. Rozpoczęcie pracy z GitHub 398
 - 14.2.2. Dodanie projektu do GitHub 399
 - 14.2.3. Współpraca przez serwis GitHub 403
- 14.3. Przekazanie własnego modułu do repozytorium npm 405
 - 14.3.1. Przygotowanie pakietu 406
 - 14.3.2. Przygotowanie specyfikacji pakietu 406
 - 14.3.3. Testowanie i publikowanie pakietu 407
- 14.4. Podsumowanie 409

Dodatek A. Instalacja Node i dodatki opracowane przez społeczność 411

- A.1. Instalacja w systemie OS X 411
 - A.1.1. Instalacja za pomocą Homebrew 412
- A.2. Instalacja w systemie Windows 413
- A.3. Instalacja w systemie Linux 414
 - A.3.1. Przygotowania do instalacji w Ubuntu 414
 - A.3.2. Przygotowania do instalacji w CentOS 415
- A.4. Kompilacja Node 415
- A.5. Używanie menedżera pakietów Node 416
 - A.5.1. Wyszukiwanie pakietów 417
 - A.5.2. Instalacja pakietu 418
 - A.5.3. Przeglądanie dokumentacji i kodu pakietu 419

Dodatek B. Debugowanie Node 421

- B.1. Analiza kodu za pomocą JSHint 421
- B.2. Dane wyjściowe debugowania 422
 - B.2.1. Debugowanie za pomocą modułu console 422
 - B.2.2. Użycie modułu debug do zarządzania danymi wyjściowymi procesu debugowania 423
- B.3. Debugger wbudowany w Node 424
 - B.3.1. Nawigacja po debuggerze 424
 - B.3.2. Analiza i zmiana stanu w debuggerze 425
- B.4. Inspektor Node 426
 - B.4.1. Uruchomienie inspektora Node 426
 - B.4.2. Nawigacja po inspektorze Node 426
 - B.4.3. Przeglądanie stanu w inspektorze Node 427

Dodatek C. Rozszerzenie i konfiguracja frameworka Express 429

- C.1. Rozszerzenie frameworka Express 429
 - C.1.1. Rejestracja szablonów silników 429
 - C.1.2. Szablony i projekt consolidate.js 430
 - C.1.3. Frameworki i rozszerzenia Express 431
- C.2. Konfiguracja zaawansowana 432
 - C.2.1. Modyfikacja odpowiedzi JSON 433
 - C.2.2. Formatowanie odpowiedzi JSON 433

Skorowidz 435