Podziękowania (11)
O autorze (13)
Wprowadzenie (15)

- 0.1. Ajax - znaczenie skrótu (16)
  - 0.1.1. Asynchroniczny (17)
  - 0.1.2. JavaScript (17)
  - 0.1.3. XML (18)
- 0.2. Cele niniejszej książki (19)
- 0.3. Wymagania wstępne potrzebne do studiowania tej książki (23)

Rozdział 1. Użyteczność (27)

- 1.1. Interfejs czy pokaz (28)
  - 1.1.1. Implementacja (30)
- 1.2. Oczekiwania użytkowników (32)
- 1.3. Wskaźniki i inne formy kontaktu z użytkownikami (34)
  - 1.3.1. Throbber (34)
  - 1.3.2. Wskaźniki postępu (37)
  - 1.3.3. Komunikaty dla użytkowników wyświetlane w pętli (40)
- 1.4. Znaczniki semantyczne (47)
  - 1.4.1. Lepsza dostępność (48)
  - 1.4.2. Łatwość użytkowania (50)
  - 1.4.3. Łatwiejsza pielęgnacja (51)
  - 1.4.4. Łatwiejsze przetwarzanie (52)
- 1.5. Co łączy style CSS z językiem JavaScript (55)

Rozdział 2. Dostępność (61)

- 2.1. Wymagania WCAG i wytyczne sekcji 508 (62)
  - 2.1.1. WCAG (63)
  - 2.1.2. Wytyczne sekcji 508 (71)
- 2.2. Czytniki ekranu mogą obsługiwać wywołania Ajax (73)
  - 2.2.1. Zastępowanie treści (74)
  - 2.2.2. Walidacja formularzy (75)
- 2.3. Dyskretny Ajax (77)
- 2.4. Projektowanie z uwzględnieniem zasad dostępności (79)
  - 2.4.1. Projektowanie aplikacji o wysokim kontraście (79)
  - 2.4.2. Interfejs z możliwością powiększania fragmentów ekranu (81)
  - 2.4.3. Kontrolki, do których łatwo dotrzeć (83)
- 2.5. WAI-ARIA (84)

Rozdział 3. Architektura aplikacji po stronie klienta (87)

- 3.1. Obiekty i wyzwalanie zdarzeń (88)
  - 3.1.1. Obsługa zdarzeń obiektów wbudowanych (90)
  - 3.1.2. Obiekty JavaScript (92)
- 3.2. Wzorzec projektowy Model-Widok-Kontroler (108)
  - 3.2.1. Model (109)
  - 3.2.2. Widok (113)

    - 3.2.3. Kontroler (122)
- 3.3. Projektowanie aplikacji sterowanych zdarzeniami (125)
    - 3.3.1. Zalety wykorzystanej architektury (126)

Rozdział 4. Debugowanie kodu po stronie klienta (129)

- 4.1. Walidacja, walidacja, walidacja (130)
    - 4.1.1. Walidator zestawu znaczników (132)
    - 4.1.2. Walidator CSS (133)
    - 4.1.3. Ekstraktor semantyczny (134)
- 4.2. Narzędzia w przeglądarkach i wtyczki (135)
    - 4.2.1. Konsola (135)
    - 4.2.2. Internet Explorer (136)
    - 4.2.3. Firefox (141)
    - 4.2.4. Opera (147)
    - 4.2.5. Safari (149)
- 4.3. Profilowanie kodu JavaScript (152)
    - 4.3.1. Rozpoznawanie "wąskich gardeł" (154)
- 4.4. Testy jednostkowe (158)
    - 4.4.1. Asercje (160)
    - 4.4.2. Konfiguracja testu (161)
    - 4.4.3. Właściwy test (164)
    - 4.4.4. Obiekty-atrapy (166)
    - 4.4.5. Zestawy testów (169)

Rozdział 5. Optymalizacja wydajności (173)

- 5.1. Wydajność bazy danych (174)
    - 5.1.1. Schemat (175)
    - 5.1.2. Zapytania (179)
- 5.2. Zajętość pasma i opóźnienia (181)
    - 5.2.1. Pasmo (183)
    - 5.2.2. Opóźnienia (187)
- 5.3. Pamięć podręczna (190)
    - 5.3.1. System plików (191)
    - 5.3.2. Pamięć (193)
    - 5.3.3. Uzupełnienie implementacji (200)
- 5.4. Wykorzystanie HTTP/1.1 (202)
    - 5.4.1. If-Modified-Since (205)
    - 5.4.2. Range (206)
- 5.5. Profilowanie kodu PHP (209)
    - 5.5.1. Debuger APD (209)
    - 5.5.2. Xdebug (213)

Rozdział 6. Skalowalne i łatwe do pielęgnacji aplikacje Ajaksa (219)

- 6.1. Ogólne praktyki (220)
    - 6.1.1. Użycie procesora (221)
    - 6.1.2. Zużycie pamięci (223)
- 6.2. Wiele prostych interfejsów (227)

    - 6.2.1. Modularność (227)
    - 6.2.2. Późne ładowanie (230)
- 6.3. Rozbudowane interfejsy (233)
    - 6.3.1. Aplikacje monolityczne (233)
    - 6.3.2. Wstępne ładowanie (237)

Rozdział 7. Architektura aplikacji po stronie serwera (239)

- 7.1. Projektowanie aplikacji dla wielu interfejsów (240)
- 7.2. Wzorzec projektowy Model-Widok-Kontroler (244)
    - 7.2.1. Model (244)
    - 7.2.2. Kontroler (254)
    - 7.2.3. Widok (263)
- 7.3. Wykorzystanie wzorca Fabryka wraz z mechanizmem obsługi szablonów (269)

Rozdział 8. Bezpieczeństwo aplikacji internetowych (275)

- 8.1. HTTPS (277)
    - 8.1.1. Po co używać HTTPS? (277)
    - 8.1.2. Bezpieczeństwo a wydajność (279)
- 8.2. Ataki typu SQL Injection (280)
    - 8.2.1. Nie używaj opcji magic_quotes (281)
    - 8.2.2. Filtrowanie (282)
    - 8.2.3. Instrukcje preparowane (284)
- 8.3. XSS (285)
    - 8.3.1. Unieszkodliwianie zestawu znaczników (286)
    - 8.3.2. Unieszkodliwianie adresów URL (291)
- 8.4. CSRF (292)
    - 8.4.1. Sprawdzanie adresu, z którego pochodzi żądanie (294)
    - 8.4.2. Przesyłanie dodatkowego nagłówka (296)
    - 8.4.3. Pomocnicze, losowe tokeny (297)
- 8.5. Nie ufaj użytkownikowi (300)
- 8.6. Nie ufaj serwerowi (302)

Rozdział 9. Dokumentacja (307)

- 9.1. Dokumentowanie kodu jest potrzebne (308)
    - 9.1.1. Odtwarzanie projektu we własnej pamięci (308)
    - 9.1.2. Ułatwienie nauki (310)
    - 9.1.3. Uważajcie na autobusy (311)
- 9.2. Dokumentowanie interfejsu API (311)
    - 9.2.1. phpDocumentor (312)
    - 9.2.2. JSDoc (320)
- 9.3. Wewnętrzna dokumentacja programisty (325)
    - 9.3.1. Standardy kodowania (327)
    - 9.3.2. Przewodniki programowania (332)
    - 9.3.3. Przewodniki stylu (333)

Rozdział 10. Projektowanie gier (335)

- 10.1. Inny rodzaj bezpieczeństwa (337)
  - 10.1.1. Walidacja (338)
  - 10.1.2. Logika serwerowej strony aplikacji (340)
- 10.2. Pojedynczy gracz (343)
  - 10.2.1. Podwójne buforowanie (343)
- 10.3. Tryb czasu rzeczywistego - wielu graczy (348)
  - 10.3.1. Odpowiedzi w postaci strumieni (349)
  - 10.3.2. Element event-source grupy WHATWG (354)
  - 10.3.3. Animacje z wykorzystaniem technik przewidywania (356)

Rozdział 11. Wnioski (359)

- 11.1. Pamiętaj o użytkownikach (360)
- 11.2. Projektuj z myślą o przyszłości (361)
- 11.3. Programuj z myślą o przyszłości (362)

Bibliografia (365)
Dodatek A: Zasoby (369)
Dodatek B: OpenAjax (373)

- Zgodność ze standardem (374)
- Rejestracja przestrzeni nazw (377)
- Zarządzanie zdarzeniami (378)

Skorowidz (381)