

Spis treści

Wprowadzenie

Dla kogo jest ta książka? (22)
Wiemy, co myślisz (23)
Wiemy, co myśli twój mózg (23)
Oto co zrobiliśmy (26)
Oto co TY możesz zrobić, aby skłonić swój mózg do posłuszeństwa (27)
Przeczytaj (28)
Zespół korektorów merytorycznych (30)
Podziękowania (31)

1. Aplikacje internetowe dla nowego pokolenia

Strony WWW: podejście tradycyjne (34)
Strony WWW: podejście nowoczesne (35)
Kiedy możemy mówić o stronie w metodologii Ajax? (37)
Rockandrollowe pamiątki Roba (38)
Ajax i rock and roll w 5 krokach (44)
Etap 2. Inicjalizacja JavaScriptu (48)
Etap 3. Tworzenie obiektu żądania (52)
Etap 4. Pobieranie informacji o przedmiocie (54)
Napiszmy kod żądający informacji o przedmiocie (56)
Zanim rozpoczniesz pracę z obiektem żądania, upewnij się, że on istnieje (57)
Obiekt żądania jest po prostu obiektem (58)
Hej, serwerze, wywołaj potem u mnie displayDetails(), dobrze? (59)
Do wysłania żądania użyj send() (60)
Na żądanie Ajaksa serwer zwykle zwraca dane (62)
Ajax jest agnostykiem serwerowym (63)
Użyj funkcji zwrotnej do pracy z danymi zwróconymi przez serwer (67)
Pobierz odpowiedź serwera z właściwości responseText obiektu żądania (68)
Żegnajcie, tradycyjne aplikacje internetowe! (70)

2. Myślenie "po ajaksowemu"

Tradycyjna witryna Mike'a jest do bani (74)
Użyjmy Ajaksa do ASYNCHRONICZNEGO przesyłania żądań rejestracji (76)
Aktualizacja strony rejestracji (81)
PROGRAMOWA konfiguracja procedury obsługi zdarzenia window.onload (84)
Kod JavaScript znajdujący się poza funkcjami jest wykonywany podczas odczytu skryptu (86)
Co kiedy się wydarza? (87)
A na serwerze... (88)
Niektóre części projektów ajaksowych będą takie same... zawsze (90)
Funkcja createRequest() jest zawsze taka sama (91)
Twórz obiekt żądania... w wielu przeglądarkach (94)
Projekt aplikacji ajaksowej obejmuje zarówno stronę WWW, jak i program po stronie serwera (96)
Co już zrobiliśmy... (99)
Co jeszcze musimy zrobić... (99)
Obiekt żądania łączy twój kod z przeglądarką (102)
Porozumiewasz się z przeglądarką, a nie z serwerem (103)
Przeglądarka wywołuje kod zwrotny i przekazuje do niego odpowiedź serwera (106)
Pokaż Mike'owi ajaksową stronę rejestracji (108)
Teraz formularz może przysyłać żądania do serwera na dwa sposoby (109)
Utwórzmy klasy CSS dla każdego stanu przetwarzania żądania... (112)
...i zmieńmy klasę CSS za pomocą JavaScriptu (113)
Zmiany? Nie potrzebujemy ich! (114)
Zezwalaj na rejestrację tylko wtedy, gdy wprowadzono odpowiednie dane (115)

3. Reagowanie na użytkowników

Wszystko zaczęło się od "psa z głową w dół" (124)
Aplikacje ajaksowe to coś więcej niż suma ich części (131)
Zdarzenia są kluczem do interaktywności (134)
Połącz zdarzenia ze strony WWW z procedurami obsługi w kodzie JavaScript (137)
Za pomocą zdarzenia `window.onload` zainicjalizuj pozostałe elementy interaktywne strony (138)
Niech przyciski po lewej stronie reagują na kliknięcia (143)
Dopisz też kod funkcji `hideHint()` (147)
Karty: złudzenie optyczne (i graficzne) (148)
Sięgaj po obrazy za pomocą pętli `for...` (149)
Klasy CSS są (znowu) kluczem do rozwiązania problemu (150)
Hm... ale karty nie są `<a>!` (151)
To popsuło nasz JavaScript, prawda? (152)
Użyj obiektu `request` do pobrania z serwera informacji o zajęciach (157)
Zachowaj ostrożność, gdy masz dwie funkcje zmieniające tę samą część strony (158)
Gdy musisz zmieniać obrazy w skrypcie, myśl o zmienianiu klas CSS (163)
Potrzebujemy też funkcji wyświetlającej i ukrywającej przycisk (165)

4. Dwoje to już towarzystwo

Do zdarzenia może być przypisana tylko jedna procedura obsługi (a przynajmniej tak się wydaje) (170)
Procedury obsługi zdarzeń są po prostu właściwościami (171)
Właściwość może mieć tylko JEDNĄ wartość (171)
Przypisz kilka procedur obsługi zdarzeń, korzystając z `addEventListener()` (172)
W DOM Poziom 2 do jednego zdarzenia na obiekcie można przypisać kilka procedur obsługi (174)
Co się dzieje z Internet Explorerem? (178)
Internet Explorer używa zupełnie innego modelu zdarzeń (179)
`attachEvent()` i `addEventListener()` są funkcjonalnymi odpowiednikami (179)
`addEventListener()` działa we WSZYSTKICH aplikacjach, nie tylko na stronie Marty (184)
Użyjmy naszej nowej funkcji narzędziowej w `initPage()` (185)
W poszukiwaniu rozwiązania użyj `alert()` (187)
Co jeszcze może być problemem? (187)
W IE właścicielem procedur obsługi jest szkielet zdarzeń IE, a NIE aktywny obiekt strony (189)
`attachEvent()` i `addEventListener()` przesyłają jeszcze jeden argument do naszych funkcji (190)
Musimy nazwać argument `Event`, aby nasze funkcje mogły z nim pracować (191)
Ty mówisz `target`, ja mówię `srcElement` (192)
A więc jak POBIERAMY obiekt, który wyzwolił zdarzenie? (196)

5. To jak odnawianie prawa jazdy

Co tak naprawdę oznacza asynchroniczność? (202)
Cały czas budowałeś aplikacje asynchroniczne (204)
Ale czasem możesz ledwo zauważyć... (205)
Mówiąc o przetwarzaniu po stronie serwera... (206)
Asynchroniczność w trzech łatwych krokach (209)
Potrzebujemy dwóch pól na hasła oraz `<div>` na zdjęcia okładek (210)
Jeżeli potrzebujesz nowego zachowania, prawdopodobnie potrzebujesz też nowej funkcji obsługującej zdarzenie (215)
Za pomocą JEDNEGO obiektu `request` możesz bezpiecznie wysłać i odebrać JEDNO żądanie asynchroniczne (224)
Żądania asynchroniczne nie czekają na nic... nawet na siebie! (225)
Jeżeli wykonujesz DWA oddzielne żądania, użyj DWÓCH osobnych obiektów (226)
Asynchroniczność oznacza, że nie możesz polegać na KOLEJNOŚCI żądań i odpowiedzi (232)
Funkcja monitorująca OBSERWUJE aplikację... SPOZA wykonywanego kodu (237)
Funkcję monitorującą wywołuj wtedy, gdy podjęcie działania MOŻE być konieczne (238)
Dzięki zmiennym stanu funkcje monitorujące wiedzą, co się dzieje (240)
A oto nasza ostatnia sztuczka (244)
Żądania synchroniczne uniemożliwiają wykonanie czegokolwiek CAŁEMU KODOWI (246)
Użyj `setInterval()`, aby to JavaScript, a nie twój kod, uruchamiał proces (249)

6. Leśnictwo na stronie WWW

Możesz zmieniać ZAWARTOŚĆ strony... (256)
...albo STRUKTURĘ strony (257)
Do reprezentowania strony przeglądarki wykorzystują obiektowy model dokumentu (258)
...a tak go widzi przeglądarka (261)
Strona jest zbiorem powiązanych obiektów (263)
Wykorzystajmy DOM do utworzenia dynamicznej aplikacji (270)
`appendChild()` dodaje do węzła nowego potomka (281)

Możesz wyszukiwać elementy według nazwy (name) lub identyfikatora (id) (282)
Czy mogę przenieść klikniętą płytkę? (286)
Możesz poruszać się po drzewie DOM, używając relacji RODZINNYCH (288)
Drzewo DOM zawiera węzły dla WSZYSTKIEGO, co znajduje się na stronie WWW (298)
Właściwość nodeName węzła tekstowego ma wartość "#text" (300)
Wygrałem? Wygrałem? (304)
Ale poważnie... wygrałem? (305)

7. Moje życzenie jest dla ciebie rozkazem

Webville Puzzles... franszyza (308)
W Woggle płytki nie są przechowywane w komórkach tabeli (312)
"Nie chcemy ZUPEŁNIE przypadkowych liter" (315)
Cała prezentacja znajduje się w CSS-ie (317)
Potrzebujemy nowej procedury obsługi zdarzenia dla kliknięć (319)
Zaczynamy tworzyć procedurę obsługi kliknięcia płytek (320)
Procedurę obsługi zdarzenia możemy przypisać w funkcji randomizeTiles() (320)
W JavaScriptcie wartości właściwości są po prostu łańcuchami (321)
Do <div> currentWord musimy dodać treść ORAZ strukturę (324)
Do zmian struktury strony użyj DOM-u (324)
Do tworzenia elementu DOM służy createElement() (325)
Musisz POWIEDZIEĆ przeglądarce, gdzie ma umieścić tworzony węzeł DOM (326)
Musimy wyłączać poszczególne płytki. To oznacza zmienianie klasy CSS płytki... (334)
...ORAZ WYŁĄCZENIE procedury obsługi zdarzenia addLetter() (334)
Przesłanie słowa jest po prostu (kolejnym) żądaniem (336)
Nasz kod JavaScript nie interesuje się tym, jak serwer opracowuje odpowiedź na nasze żądanie (336)
Test użyteczności: KIEDY wywołujemy submitWord()? (341)

8. Nie ufaj nikomu

A więc jakie frameworki SĄ dostępne (357)
W każdym frameworku obowiązuje inna składnia (358)
Składnia może być inna, ale JavaScript pozostaje niezmienny (359)
Używać frameworków czy ich nie używać? (362)
Wybór należy do ciebie... (364)

9. Więcej niż mogą wyrazić słowa

Dostosowujemy rock klasyczny do XXI wieku (366)
Jak serwer powinien przesłać odpowiedź z WIELOMA wartościami? (369)
XML sam się opisuje (388)

10. Syn JavaScriptu

JSON może być tekstem ORAZ obiektem (401)
Dane JSON można traktować jak obiekt JavaScript (402)
A więc jak pobrać dane JSON z odpowiedzi serwera? (403)
JavaScript potrafi interpretować dane tekstowe (405)
Użyj eval(), aby ręcznie interpretować tekst (405)
Interpretowanie danych JSON zwraca ich obiektową reprezentację (406)
Obiekty JavaScript są JUŻ dynamiczne... ponieważ nie są obiektami SKOMPILOWANYMI (412)
Możesz uzyskać dostęp do składników obiektu... a następnie pobrać za ich pomocą wartości z obiektu (413)
Odpowiedź serwera musisz PARSOWAĆ, a nie tylko INTERPRETOWAĆ (419)
Który format danych jest lepszy? (422)

11. Powiedz to, co chciałeś powiedzieć

Joga dla programistów... świetnie prosperujący interes (428)
Walidacja powinna przebiegać od strony WWW w kierunku serwera (434)
Możesz sprawdzać poprawność FORMATU danych i możesz sprawdzać poprawność ZAWARTOŚCI danych (440)
Musimy sprawdzić poprawność FORMATU danych z formularza zapisów (441)
Nie powtarzaj się: DRY (443)
Napiszmy jeszcze kilka procedur obsługi zdarzeń (446)
POWRÓT SYNA JavaScriptu (450)
Wartość właściwości może być kolejnym obiektem JavaScriptu (450)
Ostrzeżmy klientów Marty, gdy wystąpi błąd we wprowadzonych danych (453)
Jeżeli nie ostrzegamy, stosujemy unwarn() (457)

JEŻELI istnieje ostrzeżenie, usuń je (457)
Powtarzanie danych jest problemem SERWERA (463)
A więc skończyliśmy, prawda? (464)

12. Paranoja to twoja przyjaciółka

W filmie występuje czarny charakter (466)
W żądaniach GET parametry żądania są przesyłane jako jawny tekst (469)
Żądania typu POST NIE przesyłają jawnego tekstu (470)
Dane w żądaniu POST są ZAKODOWANE, dopóki nie znajdą się na serwerze (472)
Wysyłamy dane w żądaniu POST (474)
Zawsze upewnij się, że dane żądania zostały OTRZYMANE (476)
Dlaczego żądanie POST nie zadziało (478)
Serwer dekoduje dane POST (479)
Musisz POWIEDZIEĆ serwerowi, co wysyłasz (480)
A Pięć najważniejszych tematów (których nie omówiliśmy)
1. Inspekcja DOM-u (492)
2. Łagodna degradacja (495)
3. Biblioteki script.aculo.us oraz Yahoo UI (496)
4. Użycie bibliotek JSON w kodzie PHP (498)
5. Ajax i ASP.NET (500)
B Po prostu daj mi kod
utils.js: etapy rozwoju (504)

Skorowidz (507)