JavaScript. Podręcznik tworzenia interaktywnych stron internetowych. Wydanie II

Autor: Dave Thau
Tłumaczenie: Marcin Karbowski, Tomasz Walczak
ISBN: 978-83-246-1079-2
Tytuł oryginału: The Book of JavaScript, 2nd Edition: A Practical Guide to Interactive Web Pages
Format: B5, stron: około 500

Zaprojektuj tętniące życiem strony internetowe

- Jak umieszczać na stronach WWW elementy interaktywne?
- W jaki sposób weryfikować poprawność danych w formularzach?
- Jak najlepiej wykorzystać technologię AJAX?

JavaScript to język programowania należący do grupy najpopularniejszych narzędzi wykorzystywanych przy tworzeniu witryn internetowych. Ten łatwy w opanowaniu język interpretowany po stronie klienta jest stosowany do wielu zadań; weryfikowanie poprawności danych w formularzach, generowanie efektów graficznych, sprawdzanie modelu przeglądarki użytkownika i tworzenie dynamicznych menu to jego najczęstsze zastosowania. JavaScript jest również bazą dla zyskującej ogromną popularność technologii AJAX, która sprawia, że aplikacje sieciowe coraz bardziej upodobniają się do programów „biurkowych”

„JavaScript. Podręcznik tworzenia interaktywnych stron internetowych. Wydanie II” to kompleksowy przegląd możliwości języka JavaScript. Czytając tę książkę, poznasz podstawy języka JavaScript, metody wykrywania typu przeglądarki, sterowania ramkami i oknami przeglądarki, sprawdzania danych z formularzy oraz tworzenia map obrazkowych. Dowiesz się, jak korzystać z plików cookie i dynamicznego HTML. W dalszej części książki znajdziesz szczegółowy opis technologii AJAX. Nauczysz się projektować nowe strony w tej technologii i konwertować do niej istniejące dokumenty. Przeczytasz o implementacji mechanizmów AJAX po stronie serwera i przeglądarki, o korzystaniu ze skryptów PHP i usuwaniu błędów ze skryptów.

- Osadzanie kodu JavaScript w dokumencie HTML
- Korzystanie ze zmiennych
- Tworzenie efektów rollover
- Sterowanie oknami przeglądarki
- Obsługa formularzy HTML
- Zapisywanie danych użytkowników w plikach cookie
- Dynamiczny HTML
- Podstawy modelu AJAX
- AJAX po stronie przeglądarki i serwera
- Korzystanie z plików XML
- Debugowanie skryptów JavaScript i Ajax

Przekonaj się, jak JavaScript zmienia oblicze witryn WWW

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

7

Przesyłanie i odbieranie informacji za pomocą formularzy

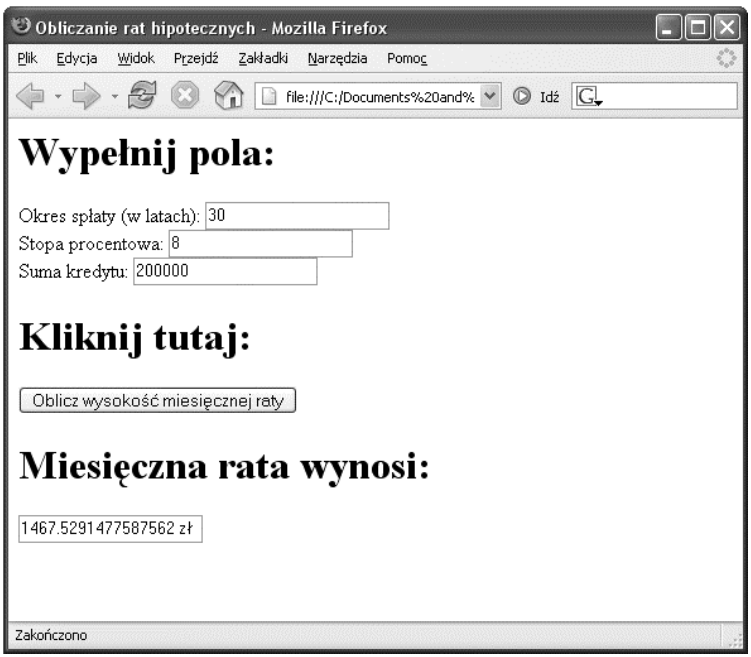
Poznaliście już kilka sposobów na pobieranie informacji od osób odwiedzających stronę. Umiecie zadawać pytania za pomocą funkcji `prompt()` i potraficie aktywować zdarzenia za pomocą uchwytów `onClick` czy `onMouseOver`. W tym rozdziale opanujecie wiele nowych technik umożliwiających pobieranie i wyświetlanie danych z użyciem formularzy HTML i języka JavaScript. Formularze i skrypty pozwalają tworzyć bardzo interaktywne strony, zawierające ankiety i quizy, kalkulatory, gry oraz nowatorskie narzędzia nawigacyjne.

Podczas lektury tego rozdziału dowiecie się, jak:

- tworzyć formularze HTML;
- odczytywać zawartość wypełnionego przez użytkownika formularza za pomocą skryptu;
- pisać skrypty automatycznie wypełniające formularze;
- używać formularzy w charakterze narzędzi nawigacyjnych.

Przykład zastosowania formularzy

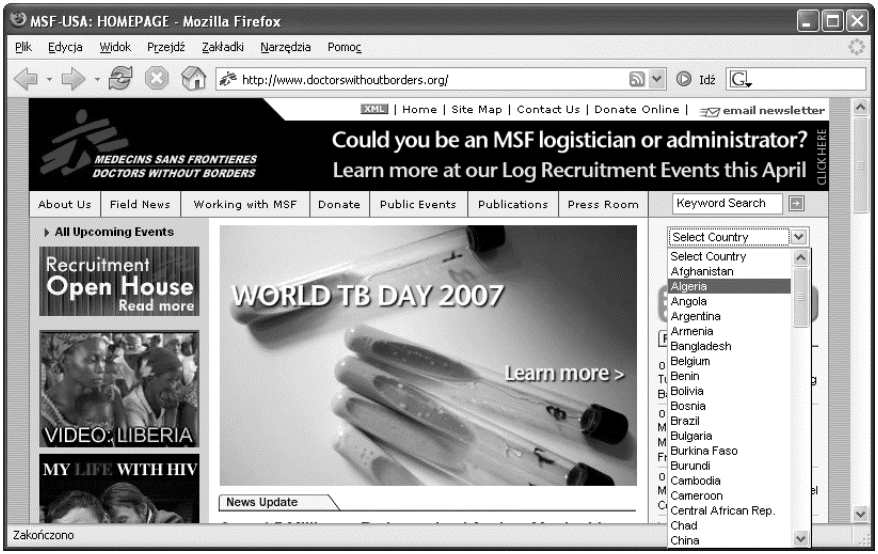
Formularze pozwalają na gromadzenie wszelkiego rodzaju informacji — danych demograficznych, takich jak wiek i płeć, odpowiedzi na pytania w quizach i sondażach, a także liczb wykorzystywanych w skomplikowanych obliczeniach. Przykładem ostatniego z wymienionych zastosowań jest przedstawiony na rysunku 7.1 kalkulator, obliczający wysokość miesięcznych rat hipotecznych. Użytkownik wpisuje sumę zaciągniętego kredytu, stopę procentową oraz okres spłaty. Po podaniu wszystkich wymienionych informacji i kliknięciu przycisku *Oblicz wysokość miesięcznej raty* skrypt pobiera dane z formularza, przeprowadza odpowiednie obliczenia i wyświetla rezultat w widocznym poniżej polu tekstowym.



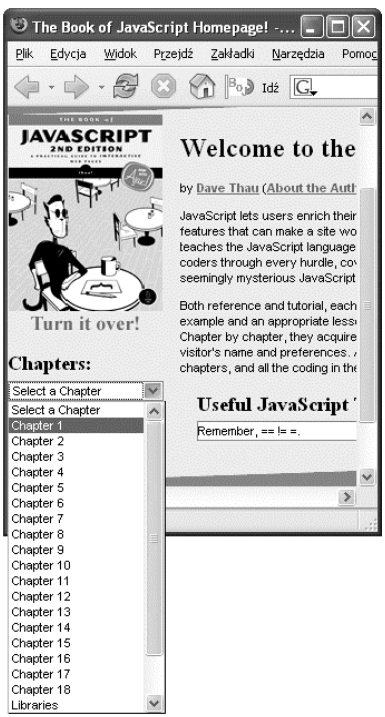
Rysunek 7.1. Formularz obliczający wysokość rat hipotecznych

Formularze można również wykorzystać w charakterze narzędzi nawigacyjnych. Na międzynarodowej stronie organizacji Lekarze bez granic (*http://www.doctors-withoutborders.org*) wykorzystano do tego celu rozwijane menu (patrz rysunek 7.2). Po wybraniu z niego nazwy kraju, o którym chcemy uzyskać więcej informacji, skrypt skieruje nas do odpowiedniej strony.

Trzeci przykład znajduje się na stronie oryginalnego wydania tej książki. Umieściłem tam rozwijane menu, które można wykorzystać do nawigacji (patrz rysunek 7.3). Po kliknięciu menu i wybraniu z niego nazwy rozdziału wyświetlona zostanie podstrona zawierająca informacje na temat danej części książki. Całość przedstawiona została na rysunku 7.3.



Rysunek 7.2. Strona domowa organizacji Lekarze bez granic, wykorzystująca rozwijane menu w charakterze narzędzia nawigacyjnego

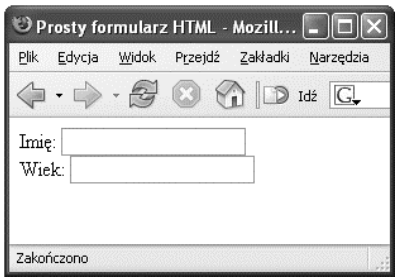


Rysunek 7.3. Element nawigacyjny na oficjalnej stronie tej książki

Wszystkie zaprezentowane elementy działają na takiej samej zasadzie: formularze opracowywane są z użyciem języka HTML, a wprowadzone do nich dane przetwarzane są przez JavaScript. Większość formularzy wykorzystujących skrypty działa na podobnej zasadzie. Naukę zaczniemy od procedury tworzenia formularzy.

Tworzenie formularzy

Na rysunku 7.4 widoczny jest prosty formularz wyświetlony w oknie przeglądarki. Poniżej (listing 7.1) przedstawiony został odpowiadający mu kod HTML.



Rysunek 7.4. Prosty formularz HTML

Listing 7.1. Kod HTML formularza widocznego na rysunku 7.4

```
<html>
<head>
<title>Prosty formularz HTML</title>
</head>
<body>
❶ <form>
❷ Imię: <input type = "text"> <br>
❸ Wiek: <input type = "text"> <br>
❹ </form>
</body>
</html>
```

Pola tekstowe

Jak widzicie na rysunku 7.4, przedstawiony kod (listing 7.1) tworzy w oknie przeglądarki dwa pola tekstowe. Osoba odwiedzająca stronę może kliknąć po kolei każde z nich i wprowadzić swoje imię oraz wiek.

Formularz tworzony jest wyłącznie za pomocą zwykłych znaczników HTML. Jak większość tego typu elementów, muszą one zostać umieszczone między znacznikami `<body>` oraz `</body>`. Początek formularza zaznaczony jest znacznikiem `<form>`, a jego koniec — znacznikiem `</form>` (wiersze ❶ i ❹). Pomiędzy tymi znacznikami znajdują się **elementy** formularza (wiersze ❷ i ❸)

przechowujące informacje. Podczas lektury tego rozdziału poznacie wiele różnych elementów formularzy, z których każdy posiada inne właściwości. Elementy zdefiniowane w wierszach ❷ i ❸ to **pola tekstowe**. Pozwalają one użytkownikowi wpisać ciąg znaków. W dalszej części rozdziału dowiecie się, w jaki sposób można pobrać i przetworzyć taki ciąg za pomocą skryptu JavaScript.

Przeglądarka rysuje pole tekstowe po napotkaniu w kodzie HTML znacznika `<input>` (wiersze ❷ i ❸):

```
<input type = "text">
```

Znacznik ten odpowiada polu pozwalającemu na wprowadzenie danych wejściowych. W tym przypadku są to dane typu tekstowego. Pole tekstowe można nieco zmodyfikować — na przykład zwiększyć jego wymiary:

```
<input type = "text" size = "40">
```

Parametr `size` jest mniej więcej równy liczbie znaków, które mogą zmieścić się w polu tekstowym.

Możliwe jest również umieszczenie w polu tekstowym wybranego ciągu znaków. Na przykład aby umieścić w pierwszym polu napis: „Tu wpisz swoje imię", należy posłużyć się następującym zapisem:

```
<input type = "text" value = "Tu wpisz swoje imię">
```

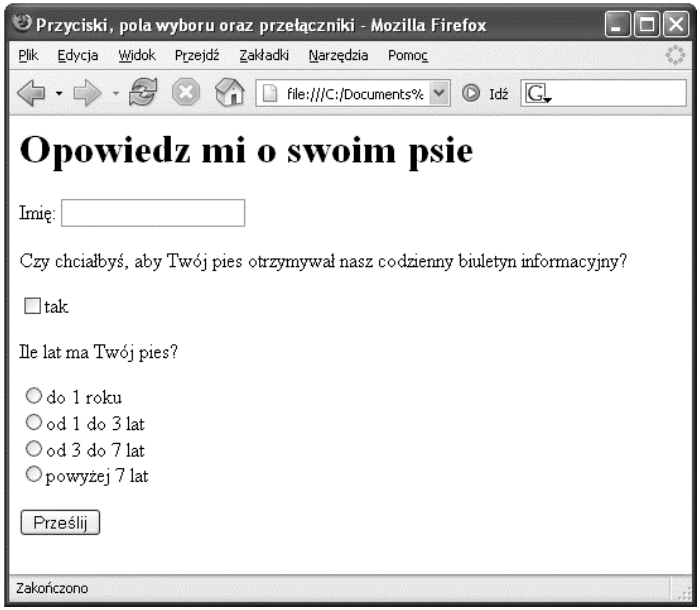
Parametr `value` pozwala zatem określić wstępną zawartość pola tekstowego. Zapamiętajcie jego nazwę — będziemy ją wykorzystywać w dalszej części rozdziału.

Przyciski, pola wyboru oraz przełączniki

Oprócz pól tekstowych w formularzach umieszczać można przyciski, pola wyboru oraz przełączniki. Wszystkie wymienione elementy zaprezentowane zostały na rysunku 7.5. Przypisany im kod znajduje się w listingu 7.2.

Listing 7.2. Pole wyboru, przełączniki i przycisk

```
<html>
<head>
<title>Przyciski, pola wyboru oraz przełączniki</title>
</head>
<body>
<h1>Opowiedz mi o swoim psie</h1>
<form>
<p>Imię: <input type = "text"/></p>
<p>Czy chciałbyś, aby Twój pies otrzymywał nasz codzienny biuletyn
informacyjny? <br/>
```



Rysunek 7.5. Przyciski, pola wyboru oraz przełączniki

```
❶ <p><input type = "checkbox"/>tak</p>
<p> Ile lat ma Twój pies? <br/>
❷ <input type = "radio" name = "age"/>do 1 roku<br/>
❸ <input type = "radio" name = "age"/>od 1 do 3 lat<br/>
❹ <input type = "radio" name = "age"/>od 3 do 7 lat<br/>
❺ <input type = "radio" name = "age"/>powyżej 7 lat<br/>
<p>
❻ <input type = "button" value = "Prześlij"/>

</form>
</body>
</html>
```

Pola wyboru

W wierszu ❶ zaprezentowanego powyżej kodu (listing 7.2) zdefiniowano pojedyncze pole wyboru. Jeśli chcecie, by po wyświetleniu strony było ono domyślnie zaznaczone, musicie wpisać słowo **checked** wewnątrz znacznika omawianego elementu w następujący sposób:

```
<input type = "checkbox" checked>
```

Termin `checked` także będzie często wykorzystywany, w związku z czym warto go zapamiętać.

Przełączniki

Kolejnym elementem formularza jest przełącznik. Przełączniki różnią się od pól wyboru tym, że stosowane są wobec grup wzajemnie wykluczających się opcji. Pies nie może **jednocześnie** mieć „mniej niż 1 rok" i „od 1 do 3 lat" — w związku z tym do zaznaczenia odpowiedzi na to pytanie przełączniki nadają się idealnie. Aby przypisać wybrane przełączniki do jednej grupy, należy nadać im taki sam atrybut `name`. W przedstawionym przykładzie (listing 7.2, wiersze od ❷ do ❺) wszystkie przełączniki mają atrybut `name` o wartości `age`. W rezultacie osoba odwiedzająca stronę będzie mogła zaznaczyć tylko jeden z nich. Na przykład jeśli użytkownik zaznaczy pierwszy przełącznik, a następnie trzeci, zaznaczenie pierwszego z nich zostanie usunięte. Podobnie jak w przypadku pola wyboru, jeden z przełączników również może zostać domyślnie zaznaczony podczas otwierania strony. Efekt ten uzyskać można, wpisując słowo **`checked`** w jego znaczniku:

```
<input type = "radio" name = "age" checked>
```

Przycisk

Ostatni z elementów przedstawionych w listingu 7.2 to przycisk:

```
<input type = "button">
```

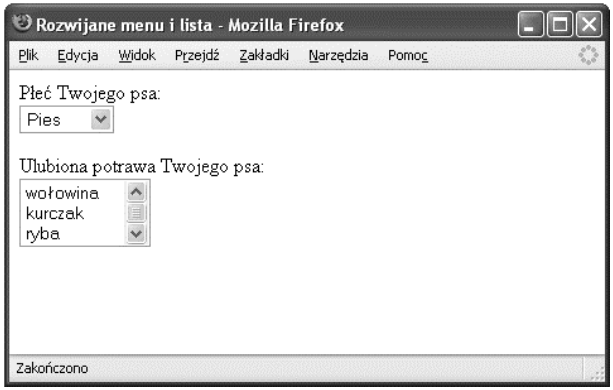
Powyższa instrukcja tworzy prostokątny przycisk. W jego wnętrzu umieścić można wybrany ciąg znaków, wpisując go jako wartość atrybutu `value` (wiersz ❻). Obecnie umieszczony na stronie przycisk nie spełnia żadnej funkcji, ale wkrótce dowiecie się, w jaki sposób da się przypisywać mu określone operacje.

Listy i rozwijane menu

Wszystkie omówione do tej pory elementy pozwalały na wprowadzanie danych wybranych przez użytkownika. Kolejne dwa elementy — lista i rozwijane menu — umożliwiają opracowanie gotowej listy wartości, spośród których osoba odwiedzająca stronę może wybrać te, które jej najbardziej odpowiadają. Oba wspomniane elementy widoczne są na rysunku 7.6. Ich kod zaprezentowany został w listingu 7.3.

Listing 7.3. Rozwijane menu i lista

```
<html>
<head>
<title>Rozwijane menu i lista</title>
</head>
<body>
<form>
Płeć Twojego psa:<br/>
```



Rysunek 7.6. Rozwijane menu i lista

```
<select>
<option>Pies</option>
<option>Suczka</option>
</select>
<p>
Ulubiona potrawa Twojego psa: <br/>
❶ <select size = "3">
❷ <option>wołowina</option>
<option>kurczak</option>
<option>ryba</option>
<option>wieprzowina</option>
<option>kot</option>
<option>sałata</option>
<option>kaktus</option>
❸ </select>

</form>
</body>
</html>
```

Rozwijane menu definiujemy między znacznikami `<select>` (wiersz ❶) i `</select>` (wiersz ❸). Każdy element takiego menu musi zostać poprzedzony znacznikiem `<option>` (wiersz ❷). Nie musicie umieszczać każdej opcji w osobnym wierszu, ale zwiększa to przejrzystość kodu.

Czasami jedna z opcji ma zostać zaznaczona jako domyślna podczas wczytywania strony. Efekt ten można osiągnąć, wpisując słowo **`selected`** wewnątrz znacznika `<option>`. Aby zdefiniować słowo `Suczka` jako wartość domyślną rozwijanego menu, możecie posłużyć się następującym zapisem:

```
<option selected>Suczka</option>
```

Podstawową różnicę między listą i rozwijanym menu tworzy parametr `size`, który można umieścić wewnątrz znacznika `<select>` pierwszego z wymienionych elementów (wiersz ❶). Określa on, ile opcji może pojawić się na liście. W przedstawionym przykładzie przypisałem mu wartość 3, w związku z czym po wyświetleniu strony widoczne są trzy pierwsze opcje. Aby zobaczyć kolejne, użytkownik musi posłużyć się paskiem przewijania widocznym po prawej stronie omawianego elementu formularza.

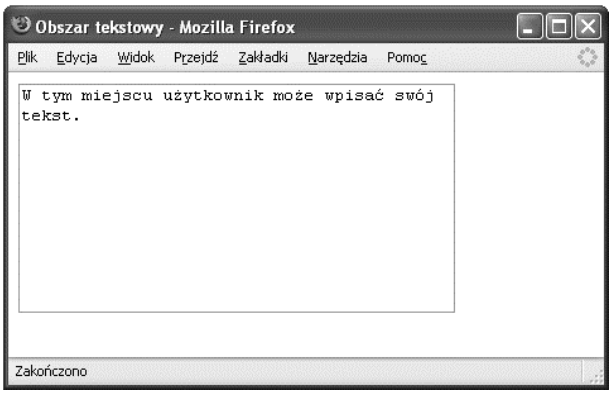
Jeśli chcecie, by osoba odwiedzająca stronę miała możliwość zaznaczenia wielu opcji na liście, umieśćcie w znaczniku `<select>` słowo `multiple`:

```
<select size = "3" multiple>
```

W ten sposób użytkownik w systemie Windows będzie mógł zaznaczać wybrane opcje, klikając je z wciśniętym klawiszem *Ctrl* (w systemie Mac OS ten sam efekt osiągnąć można, klikając opcje z wciśniętym klawiszem *Command*).

Obszar tekstowy

Możecie również zaoferować użytkownikowi możliwość wpisywania większych fragmentów tekstu. Służy do tego element określany jako **obszar tekstowy**. Dzięki niemu osoba korzystająca ze strony może wpisać dowolną ilość tekstu. Przykładowy obszar tekstowy widoczny jest na rysunku 7.7, a odpowiadający mu kod zaprezentowano w listingu 7.4.



Rysunek 7.7. Obszar tekstowy formularza

Listing 7.4. Obszar tekstowy

```
<html>
<head>
<title>Obszar tekstowy</title>
</head>
<body>
<form>
<textarea rows = "10" cols = "40">
```

```
W tym miejscu użytkownik może wpisać swój tekst.
</textarea>
</form>
</body>
</html>
```

Tekst wpisany między znacznikami `<textarea>` i `</textarea>` jest umieszczany wewnątrz obszaru tekstowego podczas wczytywania strony. Rozmiar omawianego elementu można kontrolować poprzez określenie liczby zawartych w nim wierszy i kolumn. Podobnie jak w przypadku pola tekstowego, liczba ta odpowiada z grubsza liczbie znaków możliwych do umieszczenia wewnątrz obszaru tekstowego. Atrybut `rows` definiuje wysokość opisywanego elementu, a atrybut `cols` — jego szerokość.

Podsumowanie

W tym podrozdziale pokazałem większość technik niezbędnych do tworzenia formularzy HTML wykorzystywanych w dalszej części książki. Więcej informacji na temat tych elementów stron WWW znaleźć można w każdym dobrym podręczniku do nauki języka HTML.

Formularze i JavaScript

Po umieszczeniu formularza na stronie możecie napisać skrypt pobierający wpisane w nim dane. Skrypt taki może również umieszczać określone informacje w polach formularza. Zaprezentowany na początku tego rozdziału formularz obliczający wysokość rat hipotecznych odczytywał dane wprowadzone przez użytkownika, obliczał na ich podstawie wysokość raty przypadającej na jeden miesiąc i wyświetlał wynik na stronie.

Nazywanie elementów formularza

Aby przeprowadzić dowolną operację związaną z formularzem HTML, musicie najpierw nadać odpowiednie nazwy jego elementom. Dzięki temu będziecie mogli odwoływać się do nich w ramach skryptu JavaScript. Zaprezentowany poniżej kod (listing 7.5) zawiera przykładowe znaczniki definiujące nazwę formularza (wiersz ❶) oraz nazwy jego elementów (wiersze ❷ i ❸). Znacznikowi `<option>` nie można przypisać nazwy (wiersz ❹). Strona wyświetlana po wczytaniu poniższego kodu do przeglądarki przedstawiona jest na rysunku 7.8.

Listing 7.5. Formularz z nazwami elementów

```
<html>
<head>
<title>Formularz z nazwami</title>
</head>
<body>
```

Rysunek 7.8. Formularz z nazwami elementów tworzony przez kod z listingu 7.5

```
<h1>Formularz z nazwami elementów</h1>
❶ <form name = "my_form">
❷ Imię: <input type = "text" name = "the_age_field"/><br/>
Płeć:
❸ <select name = "the_gender">
❹ <option>Mężczyzna</option>
<option>Kobieta</option>
</select>
</form>
</body>
</html>
```

Nazywając elementy formularza, należy przestrzegać tych samych zasad, które stosowane są wobec zamienianych na stronie obrazów: nie wolno wstawiać spacji, stosować nazw przypisanych już innym elementom strony i wybierać nazw zarezerwowanych dla znaczników HTML. Przykładowo polu tekstowemu nie wolno nadawać nazwy `body` ponieważ `<body>` to jeden ze znaczników języka HTML. Niektóre przeglądarki potrafią sobie co prawda poradzić z tego typu nieścisłościami, ale inne potraktują to jako błąd w skrypcie. Przyciskom, polom wyboru, obszarom tekstowym oraz przełącznikom można nadawać nazwy na takiej samej zasadzie, jak w przypadku pól tekstowych i rozwijanych menu.

Nazywanie przełączników

Nazywaniem przełączników rządzą specjalne reguły. Ponieważ wszystkie tego typu elementy należące do tej samej grupy otrzymują tę samą nazwę, nie możemy się nią posłużyć w celu wskazania jednego z nich. Można je jednak rozróżniać, przypisując wybraną wartość atrybutowi `value` (patrz listing 7.6).

Listing 7.6. Przypisywanie wartości przełącznikom

```
<html>
<head>
<title>Wartości przypisane przełącznikom</title>
```

```
</head>
<body>
<form name = "radio_button_form">
Ile lat ma Twój pies? <br/>
<input type = "radio" name = "age" value = "puppy"/>do 1 roku<br/>
<input type = "radio" name = "age" value = "young"/>od 1 do 3 lat<br/>
<input type = "radio" name = "age" value = "middle_age"/>od 3 do 7
lat<br/>
<input type = "radio" name = "age" value = "older"/>powyżej 7 lat<br>
</form>
</body>
</html>
```

Wszystkie przełączniki w powyższym kodzie posiadają taką samą wartość atrybutu `name`, dzięki czemu wiadomo, że należą one do tej samej grupy. Każdy z nich posiada jednak inną wartość atrybutu `value` — dzięki temu JavaScript potrafi zidentyfikować przełącznik zaznaczony przez użytkownika.

Nazywanie opcji

Tę samą technikę zastosować należy wobec znaczników `<option>` w elementach `<select>` formularza. Nie można im przypisywać nazw, ale możliwe jest nadawanie im wartości. Aby zatem określić, którą z opcji w rozwijanym menu zaznaczyła osoba korzystająca ze strony, musimy przypisać inną wartość każdej z nich. Przykład takiego zabiegu prześledzić można w umieszczonym poniżej kodzie (patrz listing 7.7).

Listing 7.7. Wartości wewnątrz znaczników <option>

```
<html>
<head>
<title>Formularz z wartościami przypisanymi poszczególnym opcjom</title>
</head>
<body>
<h1>Formularz z nazwami elementów</h1>
<form name = "my_form">
Wiek: <input type = "text" name = "the_age_field"/><br/>
Płeć:
❶ <select name = "the_gender">
❷ <option value = "male">Mężczyzna</option>
❸ <option value = "female">Kobieta</option>
</select>
</form>
</body>
</html>
```

W wierszu ❶ listingu 7.7 znacznikowi `<select>` przypisana została nazwa. Wartości umieszczonych w menu opcji zdefiniowane są w wierszach ❷ i ❸. Wybraną przez użytkownika opcję określić można będzie na podstawie jej atry-

butu value. Na przykład jeśli osoba korzystająca ze strony wybierze opcję Kobieta, otrzymamy wartość female przypisaną atrybutowi value w tym znaczniku <option>.

Odczytywanie i definiowanie wartości w formularzach

Po nazwaniu formularza i jego poszczególnych elementów odczytanie wybranych wartości wpisanych przez użytkownika jest bardzo proste. Wystarczy określić w skrypcie nazwę formularza oraz elementu, z którego dane mają zostać pobrane.

Odczytywanie informacji z pól tekstowych

Aby odczytać informacje zapisane w polu tekstowym o nazwie the_age_field (wiersz ❷) wykorzystanym w skrypcie z listingu 7.5, należy zastosować następujący zapis:

```
window.document.my_form.the_age_field.value
```

Powyższa instrukcja nakazuje skryptowi przeszukać obiekt window, znaleźć w nim obiekt document, a następnie umieszczony w nim formularz o nazwie my_form. W formularzu tym wyszukiwany jest element the_age_field, którego wartość zwracana jest jako wynik całego polecenia. Kod umieszczony w listingu 7.8 demonstruje, w jaki sposób można opracować prosty kalkulator przy wykorzystaniu formularza do wprowadzania danych wejściowych.

Listing 7.8. Bardzo prosty kalkulator

```
<html>
<head>
<title>Bardzo prosty kalkulator</title>
<script type = "text/javascript">
<!-- ukrywamy kod przed starszymi przeglądarkami
function multiplyTheFields()
{
❶ var number_one = window.document.the_form.field_one.value;
❷ var number_two = window.document.the_form.field_two.value;
❸ var product = number_one * number_two;
❹ alert(number_one + " razy " + number_two + " równa się " + product);
}
// od tego miejsca kod będzie z powrotem widoczny -->
</script>
</head>
<body>
❺ <form name = "the_form">
```

```
❻ Liczba 1: <input type = "text" name = "field_one"/> <br/>
❼ Liczba 2: <input type = "text" name = "field_two"/> <br/>
❽ <a href = "#" onClick = "multiplyTheFields(); return false;">Pomnóż
liczby</a>
</form>
</body>
</html>
```

W przedstawionym przykładzie wykorzystano dwa pola tekstowe oraz łącze. Po wpisaniu liczb w polach i kliknięciu łącza (patrz rysunek 7.9) wyświetlona zostaje ramka ostrzegawcza zawierająca wynik przeprowadzonych obliczeń (patrz rysunek 7.10). Kliknięcie łącza zdefiniowanego w wierszu ❽ powoduje, że wywoływana jest funkcja `multiplyTheFields()`.

Rysunek 7.9. Mnożenie dwóch liczb

Rysunek 7.10. Wyświetlanie wyniku obliczeń

Obliczenia wykonywane są przez funkcję `multiplyTheFields()`. W wierszu ❶ wartość w polu tekstowym `field_one` (wiersz ❻) formularza `my_form` (stanowiącego element obiektu `document` bieżącego okna) zapisywana jest jako zmienna `number_one`. W wierszu ❷ wykonywana jest taka sama operacja dla pola tekstowego `field_two` (wiersz ❼) — jego zawartość zapisywana jest w zmiennej `number_two`. Po odczytaniu obu liczb są one mnożone (wiersz ❸), a wynik wyświetla się w ramce ostrzegawczej (wiersz ❹).

Automatyczne wypełnianie pól formularza

Zaprezentowany na rysunku 7.9 kalkulator różni się od formularza obliczającego wysokość rat hipotecznych (rysunek 7.1) tym, że wyświetla wynik w ramce ostrzegawczej, podczas gdy wspomniany formularz podawał rezultat w polu tekstowym. Aby umieścić wybraną wartość w polu tekstowym za pomocą skryptu JavaScript, należy zapisać ją jako atrybut `value` danego pola.

Moglibyśmy zatem wzbogacić kalkulator zaprezentowany w listingu 7.8 o trzecie pole tekstowe (nazwiemy je `the_answer`) i umieścić w nim iloczyn podanych przez użytkownika liczb. W tym celu zastosować można następującą instrukcję:

```
window.document.the_form.the_answer.value = product;
```

Przypisuje ona wartość `product` atrybutowi `value` pola tekstowego `the_answer` (pole to stanowi część formularza `the_form`). Nową wersję kalkulatora obejrzeć można na rysunku 7.11. Jego kod umieszczony został poniżej (listing 7.9).

Rysunek 7.11. Umieszczanie wyniku obliczeń w polu tekstowym

Listing 7.9. Kod strony widocznej na rysunku 7.11

```
<html>
<head>
<title>Bardzo prosty kalkulator</title>
<script type = "text/javascript">
<!-- ukrywamy kod przed starszymi przeglądarkami
function multiplyTheFields()
{
    var number_one = window.document.the_form.field_one.value;
    var number_two = window.document.the_form.field_two.value;
    var product = number_one * number_two;
❶   window.document.the_form.the_answer.value = product;
}
// od tego miejsca kod będzie z powrotem widoczny -->
</script>
</head>
<body>
<form name = "the_form">
Liczba 1: <input type = "text" name = "field_one"/> <br/>
Liczba 2: <input type = "text" name = "field_two"/> <br/>
❷ Iloczyn: <input type = "text" name = "the_answer"/> <br/>
<a href = "#" onClick = "multiplyTheFields(); return false;">Pomnóż
liczby</a>
</form>
</body>
</html>
```

Ulepszona wersja kalkulatora (listing 7.9) posiada dodatkowe pole `the_answer` (wiersz ❷), w którym wyświetlany jest wynik obliczeń (wiersz ❶).

Na podstawie powyższego kodu można również wywnioskować, na jakiej zasadzie działa formularz obliczający wysokość rat hipotecznych. Nie będę zagłębiał się w szczegóły — jeśli chcecie przeanalizować przekształcenia matematyczne zastosowane we wspomnianym skrypcie, możecie go pobrać pod adresem *ftp://ftp.helion.pl/przyklady/jscpod.zip*. Nie przejmujcie się, jeśli będziecie mieli problemy z jego zrozumieniem — po przeczytaniu kolejnego rozdziału wszystko powinno wydać się Wam znacznie prostsze.

Obszary tekstowe

Obszary tekstowe (elementy pozwalające na wprowadzanie dłuższych ciągów tekstowych) obsługiwane są na takiej samej zasadzie, jak pola tekstowe. Na przykład aby wprowadzić określony ciąg znaków do obszaru tekstowego o nazwie `my_text_area` umieszczonego w formularzu `my_form`, wykorzystać można następującą instrukcję:

```
window.document.my_form.my_text_area.value =
    "Oto historia pewnej uroczej kobiety...";
```

Tekst wpisany w omawianym elemencie przez użytkownika odczytać można za pomocą poniższego polecenia:

```
var the_visitor_input = window.document.my_form.my_text_area.value;
```

Pola wyboru

Pola wyboru obsługiwane są inaczej niż pola tekstowe i obszary tekstowe. Te drugie wyróżniane są na podstawie atrybutu `value`, natomiast te pierwsze — za pomocą atrybutu `checked` (warto w tym miejscu przypomnieć sobie informacje na temat wartości logicznych przedstawione w rozdziale 3.).

Kiedy użytkownik kliknie pole wyboru, umieszczając w nim znak × lub inny rodzaj zaznaczenia, atrybut `checked` przybiera wartość `true`. Jeśli pole nie jest zaznaczone, jego atrybut `checked` ma wartość `false` (pamiętajcie — `true` i `false` to wartości logiczne, w związku z czym nie należy ich umieszczać w cudzysłowach). Wykorzystanie atrybutu `checked` zademonstrowane jest na rysunku 7.12. Odpowiedni kod znajduje się w listingu 7.10.

Listing 7.10. Quiz

```
<html>
<head>
<title>Krótki quiz</title>
<script type = "text/javascript">
<!-- ukrywamy kod przed starszymi przeglądarkami
function scoreQuiz()
```

Rysunek 7.12. Krótki quiz JavaScript

```
{
❶  var correct = 0;
❷  if (window.document.the_form.question1.checked == true) {
❸     correct = correct + 1;
   }
❹  if (window.document.the_form.question2.checked == true) {
      correct = correct + 1;
   }
❺  if (window.document.the_form.question3.checked == false) {
      correct = correct + 1;
   }
❻  alert("Udzieliłeś " + correct + " poprawnych odpowiedzi");
}
// od tego miejsca kod będzie z powrotem widoczny -->
</script>
</head>
<body>
<h1>Krótki quiz</h1>
Zaznacz prawdziwe stwierdzenia:
<form name = "the_form">
<input type = "checkbox" name = "question1"/> Mężczyźni to pozbawione
piór zwierzęta dwunożne<br/>
<input type = "checkbox" name = "question2"/> Kangury to pozbawione piór
zwierzęta dwunożne<br/>
<input type = "checkbox" name = "question3"/> Mężczyźni to kangury<br/>
<input type = "button" value = "Sprawdź wynik" onClick = "scoreQuiz();"/>
</form>
</body>
</html>
```

Kiedy użytkownik kliknie przycisk *Sprawdź wynik* umieszczony u dołu strony, wywołana zostaje funkcja `scoreQuiz()`. W wierszu ❶ kodu z listingu 7.10 tworzona jest zmienna o nazwie `correct`, której przypisywana jest wartość `0`. W zmiennej tej zapisywana jest liczba poprawnych odpowiedzi udzielonych przez użytkownika. Instrukcja w wierszach ❷ i ❸ przyznaje osobie wypełniającej quiz jeden

punkt, jeśli zaznaczyła ona pierwsze z pól. W wierszu ❷ wartość atrybutu `checked` wspomnianego pola jest pobierana i porównywana z wartością `true`. Jeśli użytkownik zaznaczył to pole, wspomniany atrybut będzie równy `true`, a zatem warunek zostanie spełniony. W rezultacie wykonane zostanie polecenie zawarte w wierszu ❸ — zwiększenie zmiennej `correct` o 1. Instrukcja w wierszu ❹ wykonuje takie same czynności w odniesieniu do drugiego pytania w quizie.

Wyrażenie `if` w wierszu ❺ różni się nieco od pozostałych dwóch. Powoduje ono zwiększenie zmiennej `correct` o 1, jeśli atrybut `checked` trzeciego pola wyboru ma wartość `false` (a więc wtedy, gdy osoba biorąca udział w quizie pozostawi to pole niezaznaczone).

Na koniec skrypt wyświetla wynik quizu (wiersz ❻).

Aby po kliknięciu przez użytkownika przycisku *Sprawdź wynik* wyświetlić poprawne odpowiedzi na pytania zamieszczone w quizie, można wykorzystać funkcję `scoreQuiz()`. Za jej pomocą należałoby określić stan każdego z pól wyboru, ustawiając ich atrybuty `checked` na wartość `true` lub `false`. W zamieszczonej poniżej wersji kodu (listing 7.11) quiz został wzbogacony o mechanizm wyświetlający właściwe odpowiedzi.

Listing 7.11. Funkcja scoreQuiz() z listingu 7.10 zmodyfikowana tak, aby wyświetlała poprawne odpowiedzi

```
function scoreQuiz()
{
   var correct = 0;
❶  if (window.document.the_form.question1.checked == true) {
      correct = correct + 1;
   } else {
❷    window.document.the_form.question1.checked = true;
   }
   if (window.document.the_form.question2.checked == true)
   {
      correct = correct + 1;
   } else {
      window.document.the_form.question2.checked = true;
   }
   if (window.document.the_form.question3.checked == false) {
      correct = correct + 1;
   } else {
      window.document.the_form.question3.checked = false;
   }
   alert("Udzieliłeś " + correct + " poprawnych odpowiedzi! Poprawne
   odpowiedzi zostały wyświetlone na ekranie.");
}
```

Do każdego wyrażenia `if` z listingu 7.11 dodana została instrukcja `else`, wstawiająca w polu poprawną odpowiedź w przypadku, gdy użytkownik popełni błąd. Polecenie w wierszu ❶ przetłumaczyć można jako: „Jeśli użytkownik zaznaczy pierwsze pole, odpowiedź na pytanie można uznać za prawidłową, w związku

z czym wartość zmiennej `correct` należy zwiększyć o 1. W przeciwnym przypadku pierwsze pole wyboru ma zostać zaznaczone w celu zasygnalizowania poprawnej odpowiedzi". Jeśli więc użytkownik odpowie niepoprawnie, pierwsze pole zostanie zaznaczone poprzez przypisanie jego atrybutowi `checked` wartości `true` (wiersz ❷).

Przełączniki

Kod odczytujący i zmieniający zaznaczenie przełączników jest nieco bardziej skomplikowany niż w przypadku pól tekstowych i pól wyboru. Wszystkie przełączniki w danej grupie mają taką samą nazwę, a więc wskazanie jednego z nich na jej podstawie staje się niemożliwe.

Aby ułatwić rozwiązanie tego problemu, JavaScript umieszcza wszystkie przełączniki o tej samej nazwie na jednej liście. Każdemu z nich przypisywany jest numer — pierwszy przełącznik ma numer 0, drugi 1, trzeci 2 itd. (większość języków programowania rozpoczyna liczenie od 0 — musicie się do tego po prostu przyzwyczaić).

Odwołanie do przełącznika tworzymy przy użyciu zapisu *`nazwa_przełącznika`*`[`*`numer_elementu`*`]`. Przykładowo, jeśli na stronie umieścicie cztery przełączniki o nazwie `age`, pierwszy z nich będzie określany jako `age[0]`, drugi jako `age[1]`, trzeci jako `age[2]`, a czwarty — jako `age[3]`.

Zaznaczenie danego przełącznika można sprawdzić za pomocą jego atrybutu `checked`, podobnie jak w przypadku pól wyboru. Załóżmy, że mamy cztery przełączniki o nazwie `age` umieszczone w formularzu `radio_button_form` (patrz listing 7.6). Aby sprawdzić, czy użytkownik zaznaczył pierwszy z nich, należy zastosować następującą instrukcję:

```
if (window.document.radio_button_form.age[0].checked == true)
{
  alert("Pierwszy przełącznik został zaznaczony!");
}
```

Metoda ta jest podobna do stosowanej wobec pól wyboru. Jedyna różnica polega na konieczności stosowania numeracji — pierwszy przełącznik w grupie `age` określany jest jako `age[0]`, podczas gdy w przypadku pól zaznaczenia wystarczyłoby podać odpowiednią nazwę.

Jeśli wie się już, w jaki sposób sprawdzić zaznaczenie przełączników, w łatwy sposób można opracować skrypt pozwalający na ich zaznaczanie. Zaznaczenie pola wyboru umożliwia poniższa instrukcja:

```
window.document.form_name.checkbox_name.checked = true;
```

W przypadku przełączników należy dodatkowo wskazać numer tego spośród nich, który ma zostać zaznaczony. Innymi słowy, aby zaznaczyć pierwszy przełącznik w grupie `age`, piszemy:

```
window.document.form_name.age[0].checked = true;
```

Rozwijane menu i listy

JavaScript daje możliwość odczytywania zawartości rozwijanych menu i list, jak również umieszczania w nich wybranych wartości na podobnej zasadzie, jak w przypadku przełączników. Istnieją jednak dwie podstawowe różnice. Po pierwsze funkcję stosowanego podczas obsługiwania przełączników atrybutu `checked` spełnia tutaj atrybut `selected`. Po drugie lista zawierająca opcje wchodzące w skład rozwijanego menu lub listy różni się od stosowanej w przypadku grupy przełączników. Z poprzedniego podrozdziału pamiętacie zapewne, że przeglądarka tworzy listę przełączników i nadaje jej nazwę zdefiniowaną w ich atrybucie `name`. W przedstawionym w listingu 7.2 przykładzie przypisałem temu atrybutowi wartość `age`, w związku z czym lista przełączników również nazwana została w ten sposób. Pierwszy element na liście nosił nazwę `age[0]`.

Rozwijane menu i listy posiadają natomiast atrybut `options` — listę wszystkich dostępnych opcji wraz z informacją, które spośród nich zostały zaznaczone. W przedstawionym poniżej (listing 7.12) kodzie pierwszym elementem (numer 0) zdefiniowanego menu jest opcja o nazwie `male`, a drugim — opcja o nazwie `female` (numer 1).

Listing 7.12. Proste rozwijane menu

```
<form name = "my_form">
<select name = "the_gender">
<option value = "male">Mężczyzna</option>
<option value = "female">Kobieta</option>
</select>
</form>
```

Poniższa instrukcja wyświetla komunikat, kiedy użytkownik zaznaczy pierwszą opcję na liście (element o nazwie `male`):

```
if (window.document.my_form.the_gender.options[0].selected == true)
{
  alert("To chłopiec!)";
}
```

Możliwe jest również automatyczne zaznaczenie jednej z opcji:

```
window.document.my_form.the_gender.options[1].selected = true;
```

Wykonanie powyższego wiersza kodu spowoduje zaznaczenie opcji `Kobieta` w rozwijanym menu.

Jeśli lista opcji w menu jest bardzo długa, a Wy chcecie jedynie zorientować się, które spośród nich zostały wybrane przez użytkownika, możecie wykorzystać atrybut `value`, przechowujący wartość każdego z zaznaczonych elementów.

W przypadku przedstawionego powyżej (listing 7.12) rozwijanego menu uzyskanie informacji na temat opcji zaznaczonej przez osobę korzystającą ze strony wymaga zapisania w kodzie poniższej instrukcji:

```
var chosen_gender = window.document.my_form.the_gender.value;
```

Gdybyście zamiast wartości danej opcji chcieli uzyskać jej numer, musielibyście wykorzystać następujący zapis:

```
var chosen_gender_index =
window.document.my_form.the_gender.selectedIndex;
```

Jeśli użytkownik zaznaczył pierwszą z dostępnych opcji, atrybut `selectedIndex` będzie miał wartość `0`.

W podrozdziale poświęconym wykorzystywaniu rozwijanych menu jako narzędzi nawigacyjnych nauczycie się zapisywać powyższe operacje w jeszcze krótszej formie. Wcześniej jednak musicie zdobyć pewne dodatkowe informacje na temat formularzy.

Obsługa zdarzeń z użyciem formularzy

Do tej pory działanie wszystkich funkcji opisywanych w niniejszym rozdziale było aktywowane przez kliknięcie łącza lub przycisku.

Każdy element posiada odrębną listę powiązanych z nim zdarzeń. Przycisk może wywoływać funkcję za pomocą uchwytu zdarzenia `onClick` w momencie, gdy zostanie naciśnięty przez użytkownika (patrz listing 7.10). Nie wszystkie elementy formularzy można jednak powiązać z tym zdarzeniem. W tabeli 7.1 przedstawione zostały niektóre zdarzenia obsługiwane przez elementy formularzy. Ich pełną listę znaleźć można w dodatku C.

Pola tekstowe, obszary tekstowe, listy oraz rozwijane menu mogą aktywować zdarzenie wyłącznie wtedy, gdy użytkownik wprowadzi w nich jakieś zmiany. Jeśli osoba wypełniająca formularz rozwinie menu i kliknie w nim już zaznaczoną opcję, zdarzenie `onChange` nie zostanie aktywowane. Analogicznie umieszczenie znaku wprowadzania tekstu wewnątrz pola tekstowego, a następnie kliknięcie w dowolnym innym punkcie strony (bez modyfikowania zawartości pola) również nie zostanie potraktowane jako zdarzenie `onChange`.

Warto również zwrócić uwagę na zdarzenie `onSubmit` obsługiwane przez formularze. Informacje zawarte w **formularzu** są przesyłane poprzez naciśnięcie klawisza *Enter* w momencie, gdy kursor wstawiania tekstu umieszczony jest w polu tekstowym, lub kliknięcie przycisku potwierdzającego wprowadzone dane. W kodzie z listingu 7.13 zdarzenie `onSubmit` zostało wykorzystane do opracowania bardzo prostej przeglądarki internetowej.

Tabela 7.1. *Elementy formularzy i obsługiwane przez nie zdarzenia*

Element formularza	Zdarzenie	Dane zdarzenie jest aktywowane przez:
Przycisk	`onClick`	kliknięcie
Pole wyboru	`onClick`	kliknięcie
Przełącznik	`onClick`	kliknięcie
Pole tekstowe	`onChange`	zmianę zawartości pola tekstowego, a następnie kliknięcie w dowolnym innym obszarze strony
Obszar tekstowy	`onChange`	zmianę zawartości obszaru tekstowego, a następnie kliknięcie w dowolnym innym obszarze strony
Lista lub rozwijane menu	`onChange`	zmianę zaznaczenia w rozwijanym menu lub liście
Formularz	`onSubmit`	naciśnięcie klawisza *Enter* w momencie, gdy kursor wstawiania tekstu umieszczony jest w polu tekstowym lub kliknięcie przycisku potwierdzającego wprowadzone dane

Listing 7.13. Uchwyt zdarzenia onSubmit wewnątrz formularza

```
<html>
<head>
<title>Prosta przeglądarka</title>
</head>
<body>
Wpisz adres URL i kliknij przycisk Wyświetl stronę lub naciśnij klawisz
Enter.
❶ <form name = "the_form" onSubmit = "window.location =
    window.document.the_form.the_url.value; return false;">
<input type = "text" name = "the_url" value = "http://" />
❷ <input type = "submit" value = "Wyświetl stronę" />
</form>
</body>
</html>
```

Operacje wykonywane w reakcji na zdarzenie `onSubmit` zdefiniowane zostały w znaczniku `<form>` (wiersz ❶). W tym przypadku instrukcja brzmi „Kiedy użytkownik potwierdzi wprowadzone dane, otwórz stronę znajdującą się pod adresem zapisanym w zmiennej `the_url`". Wybrana strona jest zatem otwierana po kliknięciu przycisku *Wyświetl stronę* lub naciśnięciu klawisza *Enter* (wiersz ❷). Wyrażenie `return false` umieszczone w wierszu ❶ zapobiega przejęciu przez przeglądarkę kontroli nad działaniem okna po przesłaniu formularza. Gdyby nie ono, polecenia JavaScript w ogóle nie zostałyby wykonane.

Skrócona postać skryptu

Znacznik <form> w wierszu ❶ listingu 7.13 jest dość długi. Aby go skrócić, można zamienić fragment kodu identyfikujący formularz window.document.the_form na słowo this. Pozwala ono tworzyć odwołania do elementu, w którym jest umieszczone. Przykładowo w wierszu ❶ zaprezentowanego powyżej kodu instrukcja pobierająca wartość zapisaną w zmiennej the_url umieszczona jest w znaczniku <form>. Oznacza, to że możecie zastąpić fragment identyfikujący ten znacznik wyrazem this. Innymi słowy, omawiane polecenie można zapisać w tak:

```
<form name = "the_form" onSubmit = "window.location =
this.the_url.value;">
```

Elementy identyfikujące formularz (window.document.the_form) zastąpiłem słowem this, ponieważ opisywany fragment kodu znajduje się wewnątrz znacznika <form>. Czasami trudno zidentyfikować obiekt, do którego odnosi się wyraz this, ale zwykle jest nim znacznik, w którym słowo to zostało wpisane.

Oto kolejny przykład. Załóżmy, że napisaliście funkcję o nazwie checkEmail(). Sprawdza ona poprawność wpisanego adresu poczty elektronicznej (zasadę jej działania omówię w rozdziale 11.). Formularz i pole tekstowe służące do wpisywania adresu generowane są przez następujący kod:

```
<form name = "the_form">
<input type = "text" name = "email" onChange =
"checkEmail(window.document.the_form.email.value);"/>
</form>
```

Elementy window.document.the_form.email umieszczone wewnątrz uchwytu zdarzenia onChange identyfikują pole tekstowe, którego część stanowi wspomniany uchwyt. Pole tekstowe przesyła zapisaną w nim wartość do funkcji checkEmail(), w związku z czym zdarzenie onChange można również zapisać tak:

```
onChange = "checkEmail(this.value);">
```

W tym przypadku słowo this zastępuje zapis window.document.the_form.email, ponieważ jest ono umieszczone w znaczniku pola tekstowego.

Rozwijane menu jako narzędzia do nawigacji

Jesteście już gotowi, aby opracować własne narzędzie nawigacyjne oparte na rozwijanym menu, podobne do umieszczonego na stronie organizacji Lekarze bez granic (patrz rysunek 7.2). Przykładowe narzędzie tego typu przedstawione zostało na rysunku 7.13. Poniżej (listing 7.14) zamieszczony jest odpowiedni skrypt.

Rysunek 7.13. Proste narzędzie nawigacyjne

Listing 7.14. Rozwijane menu jako narzędzie do nawigacji

```
<html>
<head>
<title>Nawigacja z użyciem rozwijanego menu</title>
<script type = "text/javascript">
<!-- ukrywamy kod przed starszymi przeglądarkami
function visitSite(the_site)
{
   window.location = the_site;
}
// od tego miejsca kod będzie z powrotem widoczny -->
</script>
</head>
<body>
<h1>Wybierz z menu stronę, którą chcesz wyświetlić</h1>
<form name = "the_form">
❶ <select name = "the_select" onChange =
    "visitSite(this.value);">
<option value = "http://helion.pl/">Wydawnictwo Helion</option>
<option value = "http://www.nytimes.com/">Gazeta New York Times</option>
<option value = "http://www.theonion.com/">Serwis The Onion</option>
</select>
</body>
</html>
```

Większa część widocznego powyżej (listing 7.14) skryptu powinna być dla Was zrozumiała. Jedyny bardziej skomplikowany zapis to definicja zdarzenia `onChange` w wierszu ❶ (należy pamiętać, że znaczniki `<select>` obsługują ten typ uchwytu). Kiedy następuje zdarzenie `onChange`, skrypt wywołuje funkcję `visitSite()` i przekazuje jej jako argument wartość `this.value`.

Ostatnie udoskonalenia

Czasami chcemy jedynie sprawdzić, czy zaznaczony został dany przełącznik. Jak już wiecie, efekt ten osiągnąć można za pomocą poniższej instrukcji:

```
if (window.document.radio_button_form.age[0].checked == true)
{
   alert("Zaznaczony jest pierwszy z przełączników!");
}
```

Jest ona nieco rozwlekła. Można ją jednak skrócić, dodając atrybut `id` do elementów formularza. Przykładowo:

```
<input type = "radio" name = "age" id = "age1" value = "puppy">od 0 do 1 <br>
<input type = "radio" name = "age" id = "age2" value = "young">od 1 do 3 <br>
<input type = "radio" name = "age" id = "age3" value = "middle">od 3 do 7<br>
```

Wartość tego atrybutu może być dowolna — nie wolno jedynie zdefiniować dwóch (lub więcej) takich samych `id`. Po przypisaniu każdemu z elementów formularza osobnego `id` możemy zastosować te atrybuty jako parametry metody `getElementById()`:

```
var myElement = window.document.getElementById("age1");
if (myElement.checked == true) {
   alert("Zaznaczony jest pierwszy przełącznik!");
}
```

lub skorzystać z bardziej zwięzłego zapisu:

```
if (document.getElementById("age1").checked == true) {
   alert("Zaznaczony jest pierwszy przełącznik!");
}
```

Atrybut `id` umieścić można w dowolnym elemencie HTML, nie tylko w formularzu. Jeśli znacznik `<image>` zawiera ten atrybut, możemy tworzyć do niego odwołania za pomocą tej samej metody `getElementById()`. Niektórzy wolą stosować to rozwiązanie zamiast identyfikowania elementów według ich nazw — i mają po temu dobre powody. Po pierwsze korzystając ze wspomnianej metody, można uzyskać dostęp do elementu formularza nawet wtedy, gdy nie wiemy, w którym formularzu został on umieszczony. Jest to bardzo przydatne w przypadku stron zawierających wiele formularzy. Po drugie metoda ta stanowi jedyny sposób na uzyskanie dostępu do określonych elementów HTML. Tematem tym zajmiemy się w rozdziale 13., podczas omawiania dynamicznego języka HTML.

Osobiście wolę jednak tworzyć odwołania do elementów przy użyciu ich nazw — ze względu na częstą konieczność integrowania skryptów JavaScript ze skryptami CGI (patrz rozdział 11.). Strony internetowe często przesyłają zawartość formularzy do programów działających po stronie serwera, które wykonują operacje takie, jak umieszczanie danych w bazie lub wysyłanie wiadomości do użytkowników z użyciem poczty elektronicznej. Programy te korzystają z nazw elementów, a nie ich atrybutów `id`. Aby więc uzyskać pełną funkcjonalność tworzonych formularzy, należy posługiwać się nazwami poszczególnych elementów. Jest to bardziej praktyczne rozwiązanie — zwłaszcza że opracowując formularz, i tak nadajemy jego elementom wybrane nazwy.

Analizując skrypty wykorzystywane w sieci, zauważycie, że ich autorzy często umieszczają w znacznikach zarówno atrybut `name`, jak i `id`. Oba są poprawne, a decyzja, który z nich najlepiej wykorzystać, zależy od konkretnej sytuacji i Waszych preferencji.

Sposób, w jaki działają narzędzia nawigacyjne na stronie organizacji Lekarze bez Granic

Rozwijane menu na stronie organizacji Lekarze bez granic opiera się na kodzie podobnym do zaprezentowanego w listingu 7.14. Poniżej podaję skróconą wersję skryptu zastosowanego na wspomnianej stronie (listing 7.15).

Listing 7.15. Kod rozwijanego menu na stronie organizacji Lekarze bez granic

```
<script type = "text/javascript">
<!-- ukrywamy kod przed starszymi przeglądarkami
function gotosite(site) {
❶ if (site != "") {
    self.location = site;  }
}
// od tego miejsca kod będzie z powrotem widoczny -->
</script>
<SELECT class = textbox onchange= javascript:gotosite(this.value); name =
select>
<OPTION selected>Select Country/OPTION>
<OPTION value=/news/afghanistan.cfm>Afghanistan</OPTION>
<OPTION value=/news/algeria.cfm>Algeria</OPTION>
<OPTION value=/news/angola.cfm>Angola</OPTION>
</SELECT>
```

Jedyną różnicą między powyższym kodem a skryptem nawigacyjnym przedstawionym w listingu 7.14 jest wyrażenie `if` umieszczone w wierszu ❶. Ten dodatkowy test jest konieczny, ponieważ pierwszy element na liście nie zawiera żadnej wartości — to tylko nagłówek. Jeśli użytkownik zaznaczy tę opcję, właściwość `value` przybierze wartość `""`. Instrukcja w wierszu ❶ listingu 7.15 sprawdza więc, czy osoba odwiedzająca stronę nie wybrała przypadkiem pierwszej pozycji na liście (nagłówka). Jeśli tak, funkcja nie wyświetla nowej strony.

Podsumowanie

W tym rozdziale poznaliście wiele nowych technik i elementów JavaScript i HTML. Jeśli nie do końca rozumiecie któreś z wymienionych poniżej zagadnień, przeanalizujcie jeszcze raz stosowny fragment książki. Powinniście już wiedzieć, jak:

- tworzyć formularze HTML;
- odczytywać informacje wprowadzone do formularza;
- wprowadzać wybrane przez siebie dane do formularzy;
- aktywować funkcje przy użyciu elementów formularzy;
- wykorzystywać słowo `this` do skracania nazw metod i obiektów;
- uzyskiwać dostęp do elementów HTML, korzystając z atrybutu `id` i metody `getElementById()`.

Większość formularzy składa się z wielu różnych elementów. W dodatku C omówione zostały zdarzenia, które można aktywować z użyciem poszczególnych elementów formularzy, jak również informacje, które można z nich pobrać za pomocą skryptów JavaScript.

Zadanie

Proszę stworzyć zegar wyświetlający aktualną godzinę według czasu obowiązującego w San Francisco, Nowym Yorku, Londynie oraz Tokio. Na stronie należy umieścić pole tekstowe, w którym wyświetlana będzie godzina, przycisk aktualizujący zegar oraz przełączniki odpowiadające poszczególnym strefom czasowym. Po kliknięciu wybranego przełącznika w polu tekstowym powinna się pojawić bieżąca godzina w danym mieście. Kliknięcie przycisku *Aktualizuj* powodować ma zaktualizowanie godziny w oparciu o określoną za pomocą przełączników strefę czasową. Ostateczny rezultat przedstawiony jest na rysunku 7.14.

Rysunek 7.14. Godzina w różnych strefach czasowych

Przydatne okażą się tutaj informacje na temat pozyskiwania danych dotyczących bieżącego czasu. W rozdziale 2. poznaliście polecenia pozwalające na określenie aktualnej godziny:

```
var now = new Date();
var the_hour = now.getHours();
```

Z obiektem `Date` powiązanych jest kilka metod przydatnych podczas obsługiwania różnych stref czasowych. Tutaj wykorzystać należy metody `getUTCHours()`, `getUTCMinutes()`, oraz `getUTCSeconds()`. Zwracają one jako wynik godzinę, minuty oraz sekundy bieżącego czasu według standardu UTC (ang. *Coordinated Universal Time* — uniwersalny czas koordynowany), którym zastąpiono obwiązujący wcześniej czas GMT (ang. *Greenwich Mean Time* — czas średni Greenwich).

W Londynie obowiązuje czas odpowiadający godzinie określanej przez UTC. Dla Nowego Yorku należy cofnąć zegar o pięć godzin, a dla San Francisco — o 8 godzin. Tokio wyprzedza czas UTC o 9 godzin.

Jak zwykle, odpowiedź znajdziecie w dodatku A, poświęćcie jednak temu zadaniu nieco czasu i wysiłku, zanim zajrzycie do gotowego rozwiązania — w ten sposób o wiele więcej się nauczycie. Nie jest to proste ćwiczenie — jego prawidłowe wykonanie może Wam zająć ponad godzinę.