Phil Ballard

Szybki kurs JavaScript
Wprowadzenie do języka w 24 godziny

SAMS

Helion

Tytuł oryginału: Sams Teach Yourself JavaScript in 24 Hours, Sixth Edition

Tłumaczenie: Piotr Pilch
Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki
ISBN: 978-83-283-1926-4

Authorized translation from the English language edition, entitled: JAVASCRIPT IN 24 HOURS, SAMS TEACH YOURSELF, Sixth Edition; ISBN 067233738X; by Phil Ballard; published by Pearson Education, Inc, publishing as SAMS Publishing.

Wydawnictwo HELION
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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/skjs24.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/skjs24
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

Rozdział 2

Tworzenie prostych skryptów

W tym rozdziale poznasz następujące zagadnienia:

- **Różne metody dołączania kodu JavaScript do stron internetowych.**
- **Podstawowa składnia instrukcji języka JavaScript.**
- **Deklarowanie i używanie zmiennych.**
- **Użycie operatorów matematycznych.**
- **Wstawianie komentarzy do kodu.**
- **Przechwytywanie zdarzeń myszy.**

W rozdziale 1., „Wprowadzenie do języka JavaScript", wspomniano, że JavaScript to język skryptowy zapewniający stronom internetowym większą interaktywność.

W tym rozdziale dowiesz się więcej o tym, jak kod JavaScript może być dodawany do strony internetowej. W dalszej części poznasz wybrane fundamentalne elementy składni kodu programów JavaScript, takie jak instrukcje, zmienne, operatory i komentarze. Ponadto wypróbujesz kolejne przykłady kodu.

Dołączanie kodu JavaScript do strony internetowej

W poprzednim rozdziale napisano, że programy JavaScript są przekazywane przeglądarce wraz z treścią strony. W jaki sposób jednak się to odbywa? Właściwie istnieją dwie podstawowe metody kojarzenia kodu JavaScript ze stroną HTML. Obie korzystają z elementu `<script></script>` zaprezentowanego w rozdziale 1.

Pierwsza metoda polega na dołączaniu instrukcji języka JavaScript bezpośrednio do pliku HTML, tak jak to miało miejsce w poprzednim rozdziale:

```
<script>
    ... w tym miejscu są umieszczane instrukcje języka Javascript ...
</script>
```

Druga, zazwyczaj preferowana metoda dołączania kodu polega na zapisaniu kodu JavaScript w osobnym pliku i użyciu elementu `<script>` w celu dodania tego pliku przez podanie jego nazwy w atrybucie `src` (źródło):

```
<script src='moj_kod.js'></script>
```

W powyższym przykładzie dołączany jest plik *moj_kod.js*, który zawiera instrukcje języka JavaScript. Jeśli plik JavaScript nie znajduje się w tym samym folderze co skrypt wywołujący, możesz też dodać ścieżkę (względną lub bezwzględną) do pliku:

```
<script src='/ścieżka/do/pliku/moj_kod.js'></script>
```

lub

```
<script src='http://www.przyklad.com/ścieżka/do/pliku/
↪moj_kod.js'></script>
```

Umieszczenie kodu JavaScript w osobnym pliku zapewnia kilka istotnych korzyści:

- W momencie zaktualizowania kodu JavaScript aktualizacje są natychmiast dostępne dla wszystkich stron korzystających z tego samego pliku JavaScript. Jest to szczególnie ważne w przypadku bibliotek języka JavaScript, którymi zajmiemy się w dalszej części książki.
- Kod strony HTML jest bardziej przejrzysty, a tym samym łatwiejszy do czytania i zarządzania.
- Wydajność jest nieznacznie większa, ponieważ przeglądarka buforuje dołączony plik. Oznacza to, że kopia lokalna pliku będzie dostępna w pamięci następnym razem, gdy kod będzie wymagany przez tę lub inną stronę.

Uwaga

Przyjęte jest, że tak jak w tym przykładzie, plikom z kodem JavaScript nadawane jest rozszerzenie *.js*. Dołączane pliki z kodem mogą jednak mieć dowolne rozszerzenie, a przeglądarka będzie próbować interpretować ich zawartość jako kod JavaScript.

Ostrzeżenie

Instrukcje języka JavaScript w pliku zewnętrznym NIE muszą być zawarte w znacznikach `<script> ... </script>`. Nie możesz też w pliku zewnętrznym umieszczać żadnych znaczników HTML, a jedynie sam kod JavaScript.

Listing 2.1 prezentuje użyty w rozdziale 1. kod prostej strony internetowej. Obecnie jednak do sekcji `<body>` dołączono kod JavaScript. Kod może zostać wstawiony w nagłówku lub treści strony HTML. Okazuje się, że częstsze i ogólnie zalecane jest umieszczanie kodu JavaScript w nagłówku strony, dzięki czemu zapewnia ona kilka *funkcji*, które mogą być wywoływane z dowolnego miejsca w dokumencie. O funkcjach będzie mowa w rozdziale 3., „Użycie funkcji". Na razie ograniczymy się do dodania przykładowego kodu do treści dokumentu.

Listing 2.1. Dokument HTML z dołączonym plikiem JavaScript

```
<!DOCTYPE html>
  <html>
  <head>
      <title>Prosta strona</title>
  </head>
  <body>
      <p>Treść ...</p>
      <script src='moj_kod.js'></script>
  </body>
  </html>
```

Gdy kod JavaScript zostanie dodany do treści dokumentu, instrukcje są interpretowane i wykonywane w momencie ich napotkania w trakcie renderowania strony. Po odczytaniu i wykonaniu kodu renderowanie strony jest kontynuowane do momentu zakończenia jej wyświetlania.

Wskazówka

Nie jesteś ograniczony do korzystania tylko z jednego elementu `script`. W razie potrzeby na stronie możesz umieścić dowolną ich liczbę.

Uwaga

Czasami w obrębie elementów `script` napotkasz znaczniki `<!--` i `-->` notacji komentarzy w stylu języka HTML, w których umieszczane są instrukcje języka JavaScript. Oto przykład:

```
<script>
    <!--
    ... w tym miejscu są umieszczane instrukcje języka JavaScript ...
     -->
</script>
```

Taką technikę stosowano z myślą o starszych przeglądarkach, które nie rozpoznawały znacznika `<script>`. Tego rodzaju składnia „komentarza" zapobiegała wyświetlaniu przez te przeglądarki kodu źródłowego JavaScript na ekranie wraz z treścią strony. Jeśli nie masz potrzeby wspierania bardzo starych przeglądarek, technika ta nie jest już wymagana.

Instrukcje języka JavaScript

Programy JavaScript są listami osobnych poleceń nazywanych **instrukcjami**. Aby poprawnie zinterpretować instrukcje, przeglądarka oczekuje, że każda instrukcja będzie zapisana w osobnym wierszu:

```
to jest instrukcja 1
to jest instrukcja 2
```

Alternatywnie instrukcje mogą być łączone w tym samym wierszu przez umieszczenie średnika na końcu każdej z nich:

```
to jest instrukcja 1; to jest instrukcja 2;
```

Aby jednak zwiększyć łatwość czytania kodu i ułatwić zapobieganie występowaniu trudnych do znalezienia błędów składni, dobrą praktyką jest łączenie obu metod przez umieszczenie każdej instrukcji w osobnym wierszu i zakończenie jej średnikiem:

```
to jest instrukcja 1;
to jest instrukcja 2;
```

Wstawianie komentarzy do kodu

Niektóre instrukcje nie są przeznaczone do wykonywania przez interpreter języka JavaScript przeglądarki, pozwalają natomiast twórcy wpisywać w kodzie notatki niewidoczne dla gości strony. Wiersze z takimi instrukcjami określane są mianem **komentarzy**. Istnieją konkretne reguły dotyczące dodawania komentarzy do kodu.

Komentarz zajmujący tylko jeden wiersz kodu może zostać utworzony przez umieszczenie dwóch znaków ukośnika prawego przed zawartością wiersza:

```
// To jest komentarz
```

Uwaga

Język JavaScript umożliwia również użycie jako komentarzy jednowierszowych składni komentarzy języka HTML:

```
<!-- to jest komentarz -->
```

Nie jest to jednak powszechnie stosowane w programach JavaScript.

Aby w ten sposób dodać komentarz złożony z wielu wierszy, na początku każdego z nich niezbędne jest wstawienie przedrostka:

```
// To jest komentarz
// obejmujący wiele wierszy
```

Wygodniejsza metoda wprowadzania do kodu komentarzy obejmujących wiele wierszy polega na umieszczaniu przed treścią komentarza przedrostka /*, a na jej końcu znaków */. Komentarz utworzony przy użyciu takiej składni może być złożony z wielu wierszy:

```
/*  Ten komentarz może obejmować
    wiele wierszy
    bez konieczności
    oznaczania każdego wiersza  */
```

Dodawanie komentarzy do kodu to naprawdę przydatne działanie, szczególnie podczas tworzenia większych lub bardziej złożonych aplikacji JavaScript. Komentarze mogą o czymś przypominać, a także pełnić rolę instrukcji i objaśnień dla każdej innej osoby, która później będzie się zaznajamiać z kodem.

Uwaga

Prawdą jest, że komentarze nieznacznie zwiększają wielkość pliku z kodem źródłowym JavaScript, a ponadto, że może to mieć niekorzystny wpływ na czasy ładowania stron. Generalnie jednak różnica w czasie jest tak znikoma, że będzie ledwie zauważalna. Jeśli jednak naprawdę ma to znaczenie, zawsze możesz usunąć wszystkie komentarze z wersji „produkcyjnej" pliku JavaScript, czyli wersji, która zostanie użyta w przypadku produkcyjnych witryn internetowych, a nie ich wersji rozwojowych.

Zmienne

Zmienna może być traktowana jako nazwany „segregator", w którym utrzymywana jest określona porcja danych. Takie dane mogą przyjmować wiele różnych form: liczby całkowitej lub dziesiętnej, łańcucha znaków lub innych różnych typów danych omawianych w dalszej części rozdziału i w kolejnych rozdziałach. Używanym zmiennym mogą być nadawane naprawdę dowolne nazwy, pod warunkiem że będą w nich stosowane wyłącznie znaki alfanumeryczne, znak dolara $ lub znaki podkreślenia.

Załóżmy, że istnieje zmienna o nazwie `netPrice`. Przechowywaną w niej wartość można ustawić za pomocą prostej instrukcji:

```
netPrice = 8.99;
```

Uwaga

> W języku JavaScript rozróżniana jest wielkość znaków. Zmienna o nazwie `mypetcat` to inna zmienna niż zmienna `Mypetcat` lub `MYPETCAT`.
>
> Wielu programistów używających języka JavaScript oraz innych języków programowania w przypadku nazw zmiennych lubi korzystać z tzw. konwencji ***CamelCase*** (inne określenia to *mixedCase*, *BumpyCaps* itp.). W tej konwencji złożone słowa lub frazy zawierają elementy łączone bez spacji. Pierwsza litera każdego elementu jest dużą literą, z wyjątkiem pierwszej litery całego łańcucha, która może być duża lub mała. W przytoczonym przykładzie zmienna będzie mieć nazwę `MyPetCat` lub `myPetCat`.

Operacja jest określana mianem **przypisywania wartości** zmiennej. Zauważ, że przed przypisaniem wartości nie ma potrzeby deklarowania faktu istnienia tej zmiennej, inaczej niż jest to w niektórych innych językach programowania. Język JavaScript pozwala jednak na takie działanie, oferując słowo kluczowe `var`. W większości sytuacji jest to zalecana praktyka programistyczna:

```
var netPrice;
netPrice = 8.99;
```

Alternatywnie możesz w wygodny i czytelny sposób połączyć te dwie instrukcje w jedną:

```
var netPrice = 8.99;
```

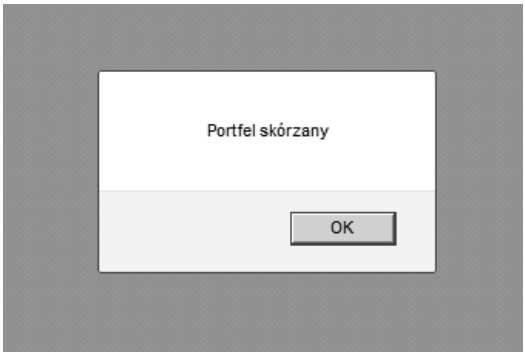
W celu przypisania łańcucha znaków jako wartości zmiennej konieczne jest ujęcie go w znaki pojedynczego lub podwójnego cudzysłowu:

```
var productName = "Portfel skórzany";
```

Możesz następnie utworzyć na przykład wiersz kodu wysyłający metodzie `window.alert` wartość zawartą w tej zmiennej:

```
alert(productName);
```

Wygenerowane okno dialogowe określi wartość zmiennej i wyświetli ją (tym razem w przeglądarce Mozilla Firefox), co pokazano na rysunku 2.1.



RYSUNEK 2.1. Wyświetlanie wartości zmiennej productName

Wskazówka

> Wybierz czytelne nazwy zmiennych. Użycie nazw, takich jak `productName` (nazwaProduktu) i `netPrice` (cenaNetto), znacznie zwiększa czytelność kodu i możliwości zarządzania nim niż w sytuacji, gdyby te same zmienne miały nazwy `var123` i `myothervar49` (nawet pomimo tego, że te dwie nazwy są całkowicie poprawne).

Operatory

Wartości zapisane w zmiennych nie będą zbyt przydatne, jeśli nie będzie możliwe modyfikowanie ich w obliczeniach.

Operacje arytmetyczne

Język JavaScript umożliwia głównie wykonywanie operacji z wykorzystaniem standardowych operatorów arytmetycznych operacji dodawania, odejmowania, mnożenia i dzielenia.

```
var theSum = 4 + 3;
```

Jak mogłeś się domyślić, po wykonaniu tej instrukcji zmienna `theSum` będzie zawierać wartość 7. W operacjach mogą być też stosowane nazwy zmiennych:

```
var productCount = 2;
var subtotal = 14.98;
var shipping = 2.75;
var total = subtotal + shipping;
```

Język JavaScript pozwala na realizowanie operacji odejmowania (`-`), mnożenia (`*`) i dzielenia (`/`) w podobny sposób:

```
subtotal = total – shipping;
var salesTax = total * 0.15;
var productPrice = subtotal / productCount;
```

Aby obliczyć resztę z dzielenia, możesz użyć operatora **dzielenia modulo** języka JavaScript. Jest on reprezentowany przez znak `%`:

```
var itemsPerBox = 12;
var itemsToBeBoxed = 40;
var itemsInLastBox = itemsToBeBoxed % itemsPerBox;
```

W przykładzie po zakończeniu ostatniej instrukcji zmienna `itemsInLastBox` zawierałaby liczbę `4`.

Język JavaScript oferuje też wygodne w użyciu operatory inkrementacji (`++`) lub dekrementacji (`--`) wartości zmiennej:

```
productCount++;
```

Odpowiada to następującej instrukcji:

```
productCount = productCount + 1;
```

Podobnie instrukcja:

```
items--;
```

jest dokładnie taka sama jak instrukcja:

```
items = items – 1;
```

Wskazówka

Jeśli musisz dokonać inkrementacji lub dekrementacji zmiennej o wartość inną niż jeden, język JavaScript również pozwala na łączenie innych operatorów arytmetycznych z operatorem = (np. += i -=).

Następujące dwa wiersze kodu są równorzędne:

```
total = total + 5;
total += 5;
```

co oznacza, że dotyczy to także następujących dwóch wierszy:

```
counter = counter - step;
counter -= step;
```

Notacji takiej możesz użyć dla innych operatorów arytmetycznych, takich jak operator mnożenia i dzielenia:

```
price = price * uplift;
price *= uplift;
```

W dodatku B, „Krótki przegląd elementów języka JavaScript", znajduje się obszerniejsza lista operatorów arytmetycznych języka JavaScript.

Pierwszeństwo operatorów

Gdy korzystasz z kilku operatorów w ramach tego samego obliczenia, w języku JavaScript stosowane są **reguły pierwszeństwa** w celu ustalenia, w jakiej kolejności obliczenie powinno być wykonane. Dla przykładu przyjrzyj się następującej instrukcji:

```
var average = a + b + c / 3;
```

Jeśli, zgodnie z tym, na co wskazuje nazwa zmiennej, próbujesz obliczyć średnią, powyższy kod nie zapewni żądanego wyniku. Operacja dzielenia zostałaby przeprowadzona dla zmiennej `c` przed dodaniem wartości zmiennych `a` i `b`. W celu poprawnego wyliczenia średniej konieczne byłoby dodanie do instrukcji nawiasu okrągłego w następujący sposób:

```
var average = (a + b + c) / 3;
```

Jeśli nie masz pewności co do reguł pierwszeństwa w obliczeniach, zawsze śmiało korzystaj z nawiasów okrągłych. To nic nie kosztuje, a dzięki temu kod staje się czytelniejszy (zarówno dla jego twórcy, jak i każdej innej osoby, która później musi modyfikować kod lub zorientować się w jego działaniu). Ponadto w ten sposób zapewnia się, że problemy z pierwszeństwem nie spowodują niepoprawnych obliczeń.

Uwaga

Jeżeli masz doświadczenie programistyczne związane z innym językiem, takim jak PHP lub Java, zapewne zauważysz, że reguły pierwszeństwa w języku JavaScript są prawie takie same jak używane w tych językach. Szczegółowe informacje o regułach pierwszeństwa języka JavaScript są dostępne pod adresem *http://msdn.microsoft.com/en-us/library/z3ks45k7(v=vs.94).aspx*.

Użycie operatora + z łańcuchami

Operatory arytmetyczne nie mają większego sensu, jeśli przetwarzane przez nie zmienne zawierają łańcuchy, a nie wartości liczbowe. Wyjątkiem jest operator +, który w języku JavaScript jest interpretowany jako instrukcja scalająca (łączenie ze sobą kolejnych elementów) dwa łańcuchy lub większą ich liczbę:

```
var firstname = "Jan";
var surname = "Nowak";
var fullname = firstname + " " + surname;
// zmienna fullname zawiera teraz wartość "Jan Nowak"
```

Jeśli spróbujesz zastosować operator + dla dwóch zmiennych, z których jedna jest łańcuchem, a druga liczbą, interpreter języka JavaScript przekształci wartość liczbową w łańcuch i połączy oba łańcuchy:

```
var name = "Dawid";
var age = 45;
alert(name + age);
```

Na rysunku 2.2 pokazano wynik użycia operatora + dla łańcucha i wartości liczbowej.

RYSUNEK 2.2. Scalanie łańcucha i wartości liczbowej

W tym miejscu jedynie ogólnie prezentujemy typy danych języka JavaScript i operacje na łańcuchach. Znacznie więcej na ten temat zamieszczono w rozdziale 5, „Liczby i łańcuchy”.

▼ Spróbuj sam!

Konwersja stopni Celsjusza na stopnie Fahrenheita

Aby dokonać konwersji temperatury wyrażonej w stopniach Celsjusza na mierzoną w stopniach Fahrenheita, musisz wykonać operację mnożenia przez 9, dzielenia przez 5, a następnie do wyniku dodać liczbę 32. Wykonajmy te działania za pomocą kodu JavaScript:

```
var cTemp = 100;  // temperatura w stopniach Celsjusza
// Śmiało korzystajmy z nawiasów okrągłych
var hTemp = ((cTemp * 9) /5 ) + 32;
```

Okazuje się, że w powyższym obliczeniu możliwe jest pominięcie wszystkich nawiasów okrągłych. Pomimo tego kod w dalszym ciągu działałby znakomicie:

```
var hTemp = cTemp*9/5 + 32;
```

Nawiasy okrągłe ułatwiają jednak zrozumienie kodu i są pomocne przy zapobieganiu błędom związanym z kwestią pierwszeństwa operatorów.

Przetestujmy na stronie internetowej kod z listingu 2.2.

Listing 2.2. Obliczanie stopni Fahrenheita na podstawie stopni Celsjusza

```
<!DOCTYPE html>
  <html>
  <head>
      <title>Konwersja do stopni Fahrenheita ze stopni
      ↪Celsjusza</title>
  </head>
  <body>
      <script>
          var cTemp = 100;  // temperatura w stopniach Celsjusza
          // Śmiało korzystajmy z nawiasów okrągłych
          var hTemp = ((cTemp * 9) /5 ) + 32;
          document.write("Temperatura w stopniach Celsjusza:
          ↪" + cTemp + " stopni<br/>");
          document.write("Temperatura w stopniach Fahrenheita:
          ↪" + hTemp + " stopni");
      </script>
  </body>
  </html>
```

Zapisz kod jako plik *temperatura.html* i załaduj go w przeglądarce. Powinien zostać uzyskany wynik widoczny na rysunku 2.3.

RYSUNEK 2.3. Dane wyjściowe kodu z listingu 2.2

Zmodyfikuj plik kilkakrotnie w celu użycia różnych wartości dla zmiennej `cTemp` i sprawdź, czy wszystko działa poprawnie.

Przechwytywanie zdarzeń myszy

Jednym z zasadniczych celów języka JavaScript jest ułatwienie zapewnienia stronom internetowym większej interaktywności związanej z użytkownikiem. Aby to osiągnąć, niezbędne są mechanizmy wykrywające, jakie działania w dowolnym danym momencie wykonuje użytkownik i program — gdzie w oknie przeglądarki znajduje się kursor myszy, czy użytkownik kliknął przycisk myszy lub nacisnął klawisz klawiatury, czy strona została całkowicie załadowana w przeglądarce itp.

Wszystkie te sytuacje są określane mianem **zdarzeń**. Język JavaScript oferuje różne narzędzia, które ułatwiają pracę ze zdarzeniami. Przyjrzyjmy się niektórym opartym na kodzie JavaScript sposobom wykrywania działań podejmowanych przez użytkownika za pomocą myszy.

Język JavaScript obsługuje zdarzenia za pomocą tzw. **procedur obsługi zdarzeń**. Omówimy trzy takie procedury: `onClick`, `onMouseOver` i `onMouseOut`.

Procedura obsługi zdarzeń onClick

Procedura obsługi zdarzeń `onClick` może być stosowana do niemal wszystkich elementów HTML widocznych na stronie. Jeden ze sposobów implementacji procedury polega na dodaniu do elementu HTML jeszcze jednego atrybutu:

```
onclick=" ...kod JavaScript... "
```

Uwaga

> Choć dodawanie procedur obsługi zdarzeń bezpośrednio do elementów HTML jest jak najbardziej dozwolone, obecnie nie jest uważane za zalecaną praktykę programistyczną. Takie rozwiązanie dobrze sprawdza się w przykładach zamieszczonych w części pierwszej książki, natomiast w jej dalszych rozdziałach poznasz bardziej eleganckie i oferujące większe możliwości sposoby używania procedur obsługi zdarzeń.

Przyjrzyjmy się przykładowi z listingu 2.3.

Listing 2.3. Użycie procedury obsługi zdarzeń onClick

```
<!DOCTYPE html>
  <html>
  <head>
      <title>Demonstracja działania procedury onClick</title>
  </head>
  <body>
        <input type="button" onclick="alert('Kliknięto przycisk!')
        ↪" value="Kliknij mnie" />
  </body>
  </html>
```

Kod HTML dodaje do elementu `<body>` strony przycisk, a ponadto zapewnia mu atrybut `onclick`. Wartość nadana temu atrybutowi jest kodem JavaScript, który ma zostać uruchomiony w momencie kliknięcia elementu HTML (w tym przypadku przycisku). Gdy użytkownik kliknie przycisk, aktywowane jest *zdarzenie* `onclick` (zwykle jest mowa o wyzwalaniu zdarzenia), po czym wykonywane są instrukcje języka JavaScript wyszczególnione w wartości atrybutu.

W tym przypadku jest to tylko jedna instrukcja:

```
alert('Kliknięto przycisk!')
```

Na rysunku 2.4 pokazano wynik kliknięcia przycisku.

RYSUNEK 2.4. Użycie procedury obsługi zdarzeń onClick

Uwaga

Być może zauważyłeś, że wywoływana jest procedura obsługi zdarzeń `onClick`, ale podczas dodawania do elementu HTML jej nazwa jest jednak zapisywana małymi literami (`onclick`). Taka konwencja wynikła stąd, że wprawdzie w języku HTML nie jest rozróżniana wielkość znaków, ale w języku XHTML takie *rozróżnienie* jednak istnieje. Język XHTML wymaga, aby nazwy wszystkich elementów i atrybutów HTML były zapisywane małymi literami.

Procedury obsługi zdarzeń onMouseOver i onMouseOut

Gdy wymagane jest jedynie wykrycie, gdzie względem konkretnego elementu strony na ekranie znajduje się wskaźnik myszy, pomocne w tym mogą okazać się procedury obsługi zdarzeń `onMouseOver` i `onMouseOut`.

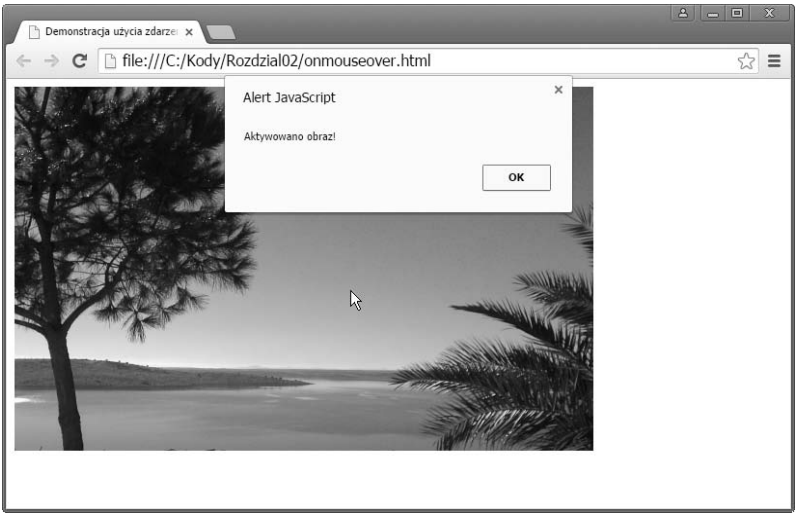
Zdarzenie `onMouseOver` jest wyzwalane w momencie umieszczenia przez użytkownika kursora myszy w obszarze ekranu zajmowanym przez żądany element. Zdarzenie `onMouseOut`, jak z pewnością już się domyśliłeś, wyzwalane jest w chwili opuszczenia przez kursor tego samego obszaru.

Listing 2.4 prezentuje prosty przykład zastosowania zdarzenia `onMouseOver`.

Listing 2.4. Użycie zdarzenia onMouseOver

```
<!DOCTYPE html>
  <html>
  <head>
      <title>Demonstracja użycia zdarzenia onMouseOver</title>
  </head>
  <body>
        <img src="image1.png" alt="image 1" onmouseover=
        ↪"alert('Aktywowano obraz!')" />
  </body>
  </html>
```

Na rysunku 2.5 pokazano wynik uruchomienia skryptu. Oczywiście zastąpienie w kodzie zdarzenia `onmouseover` zdarzeniem `onmouseout` spowoduje po prostu wyzwolenie procedury obsługi zdarzeń, a tym samym wyświetlenie okna dialogowego alertu, gdy kursor myszy *opuści* obszar obrazu, zamiast się w nim pojawić.



RYSUNEK 2.5. Użycie procedury obsługi zdarzeń onMouseOver

▼ Spróbuj sam!

Tworzenie efektu zastępowania obrazu

Za pomocą zdarzeń `onMouseOver` i `onMouseOut` możliwa jest zmiana sposobu pojawienia się obrazu, gdy zostanie na nim umieszczony kursor myszy. Aby to zrealizować, zdarzenia `onMouseOver` używamy do zmiany atrybutu `src` elementu HTML `<img>` w momencie pojawienia się kursora w jego obrębie. Z kolei zdarzenie `onMouseOut` umożliwia ponowną zmianę tego atrybutu, gdy kursor myszy opuści obszar elementu. Listing 2.5 prezentuje kod.

Listing 2.5. Efekt zastępowania obrazu za pomocą zdarzeń onMouseOver i onMouseOut

```
<!DOCTYPE html>
  <html>
  <head>
      <title>Demonstracja użycia zdarzenia OnMouseOver</title>
  </head>
  <body>
      <img src="tick.gif" alt="tick" onmouseover="this.src=
      ↪'tick2.gif';"
  onmouseout="this.src='tick.gif';" />
  </body>
  </html>
```

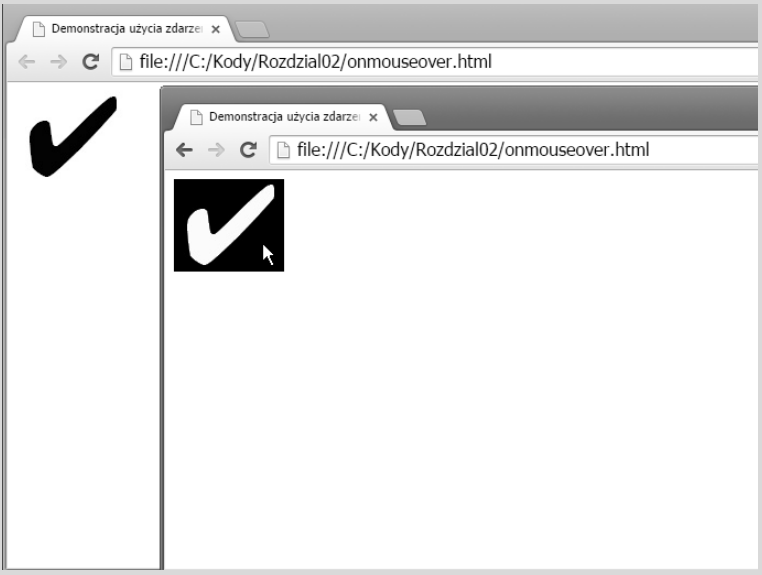
Być może zauważyłeś coś nowego w zastosowanej tutaj składni. Wewnątrz instrukcji języka JavaScript powiązanych ze zdarzeniami `onMouseOver` i `on↪MouseOut` użyto słowa kluczowego `this`.

Gdy słowo `this` zostanie użyte w obrębie procedury obsługi zdarzeń dodanej za pośrednictwem atrybutu elementu HTML, odwołuje się ono jedynie do samego elementu. W tym przypadku słowo `this` może być interpretowane jako „ten obraz". Kod `this.src` (przedstawioną notację ze znakiem kropki zastosowano już wcześniej) odwołuje się do właściwości `src` (źródło) obiektu obrazu.

W przykładzie użyto dwóch obrazów (pliki *tick.gif* i *tick2.gif*). Możesz zastosować dowolne obrazy, którymi dysponujesz, ale do celów demonstracyjnych najlepsze będą obrazy niezbyt duże i o takiej samej wielkości.

Za pomocą edytora utwórz plik HTML zawierający kod z listingu 2.5. Nazwy plików obrazów *tick.gif* i *tick2.gif* możesz zmienić na nazwy własnych dwóch plików obrazów, jeśli są one inne. Zadbaj jedynie o to, aby obrazy zostały zapisane w tym samym folderze co plik HTML. Zapisz ten plik i otwórz go w przeglądarce.

Powinno być widoczne, że obraz zmienia się w momencie umieszczenia na nim kursora myszy, po czym zmienia się ponownie w chwili opuszczenia przez kursor obszaru obrazu (rysunek 2.6).



RYSUNEK 2.6. Efekt zastępowania obrazu za pomocą zdarzeń onMouseOver i onMouseOut

Uwaga

Dawniej efekt zastępowania obrazu był osiągany zazwyczaj w ten właśnie sposób. Obecnie jednak taki efekt może być uzyskiwany znacznie szybciej za pomocą arkuszy CSS (*Cascading Style Sheet*). Niemniej jednak nadal jest to wygodny sposób demonstrowania użycia procedur obsługi zdarzeń `onMouseOver` i `onMouseOut`.

Podsumowanie

W rozdziale zaprezentowano dość dużo podstawowych informacji.

Przede wszystkim poznałeś różne metody dołączania kodu JavaScript do stron HTML.

Dowiedziałeś się, jak deklarować zmienne w kodzie JavaScript, przypisywać im wartości oraz modyfikować je za pomocą operatorów arytmetycznych.

Na końcu dokonano wprowadzenia do procedur obsługi zdarzeń języka JavaScript, a ponadto wyjaśniono, jak wykrywać określone działania wykonywane przez użytkownika za pomocą myszy.

Pytania i odpowiedzi

P. W niektórych listingach i fragmentach kodu w tym samym wierszu wyszczególniono znaczniki otwierające i zamykające `<script>`. Innym razem znaczniki umieszczono w osobnych wierszach. Czy ma to znaczenie?

O. W języku JavaScript puste odstępy, takie jak znak spacji, tabulatory i puste wiersze, są całkowicie ignorowane. Takie puste odstępy, które przez programistów są zwykle nazywane **białymi znakami**, możesz stosować do określania układu kodu w taki sposób, aby był bardziej czytelny i łatwy do prześledzenia.

P. Czy mogę użyć tego samego elementu `<script>` zarówno do dołączenia pliku zewnętrznego JavaScript, jak i do przechowywania instrukcji języka JavaScript?

O. Nie. Jeśli korzystasz z elementu `script`, aby dołączyć plik zewnętrzny JavaScript za pomocą atrybutu `src`, nie możesz już umieszczać instrukcji języka JavaScript między znacznikami `<script>` i `</script>`, ponieważ ten obszar musi pozostać pusty.

Warsztaty

Spróbuj odpowiedzieć na wszystkie pytania przed przeczytaniem zamieszczonych dalej odpowiedzi.

Quiz

1. Procedura obsługi zdarzeń `onClick` to:
   - **a)** Obiekt wykrywający położenie kursora myszy w przeglądarce.
   - **b)** Skrypt wykonywany w odpowiedzi na kliknięcie przycisku myszy przez użytkownika.
   - **c)** Element HTML, który użytkownik może kliknąć.

2. Ile elementów `<script>` dozwolonych jest na stronie?
   a) Żaden.
   b) Dokładnie jeden.
   c) Dowolna liczba.
3. Które stwierdzenie dotyczące zmiennych NIE jest prawdziwe?
   a) W nazwach zmiennych jest rozróżniana wielkość znaków.
   b) Zmienne mogą zawierać dane liczbowe i nieliczbowe.
   c) Nazwy zmiennych mogą zawierać spacje.

Odpowiedzi

1. b. Procedura obsługi zdarzeń `onClick` to skrypt wykonywany w momencie kliknięcia przycisku myszy przez użytkownika.
2. c. Możesz użyć dowolnej żądanej liczby elementów `<script>`.
3. c. W języku JavaScript nazwy zmiennych nie mogą zawierać spacji.

Ćwiczenia

- Począwszy od listingu 2.4, z elementu `<img>` usuń procedury obsługi zdarzeń `onMouseOver` i `onMouseOut`. Zamiast nich dodaj procedurę obsługi zdarzeń `onClick`, aby dla właściwości `title` obrazu ustawić łańcuch `Mój nowy tytuł` (wskazówka: dostęp do tytułu obrazu możesz uzyskać za pomocą odwołania `this.title`).
- Zaproponuj prosty sposób sprawdzenia, czy skrypt poprawnie ustawił nowy tytuł obrazu.

JavaScript to rozwiązanie odpowiednie dla programistów będących na różnych poziomach zaawansowania. Jest świetny dla początkujących — osoby, które swoją przygodę z programowaniem rozpoczęły od nauki tego języka, odkrywają, że uzyskana wiedza jest bardzo przydatna przy kodowaniu w C, Javie czy PHP. Razem z JavaScriptem można również stosować programowanie obiektowe, umożliwia on korzystanie z modelu DOM czy też używanie zewnętrznych bibliotek.

Sięgnij po tę książkę i zacznij programować już dziś! Po lekturze jej 24 rozdziałów — czytanie każdego zajmie najwyżej godzinę — poznasz podstawy programowania w języku JavaScript i szybko zaczniesz tworzyć skrypty wzbogacające strony internetowe w efektowne funkcje. Opanujesz podstawowe i bardziej złożone paradygmaty programowania, nauczysz się tworzenia i obsługiwania obiektów, zaznajomisz się z aspektami profesjonalnego projektowania aplikacji, takimi jak: sprawdzone praktyki tworzenia kodu, debugowanie kodu JavaScript i testowanie jednostkowe. Co istotne, cała praca odbywać się będzie zgodnie z aktualnymi standardami internetowymi!

Dowiedz się, jak:

- **Budować dynamiczne i interaktywne skrypty obsługiwane przez wszystkie przeglądarki**
- **Pisać przejrzysty i niezawodny kod wielokrotnego użytku**
- **Stosować techniki programowania obiektowego**
- **Wykorzystać popularną bibliotekę jQuery i kontrolować arkusze stylów CSS za pomocą prostego kodu JavaScript**
- **Rozpocząć pracę z takimi środowiskami jak AngularJS**
- **Budować dodatki i rozszerzenia przeglądarek**

Phil Ballard — doradca w branży technologii internetowych, specjalizuje się w projektowaniu stron WWW, mechanizmach SEO, tworzeniu skryptów serwerowych, projektowaniu aplikacji. Napisał wiele książek, w tym *AJAX w mgnieniu oka* — publikację wydaną przez Helion.

Helion

SAMS

38936 numer katalogowy

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-283-1926-4

9 788328 319264

Informatyka w najlepszym wydaniu

cena: 69,00 zł