Instrukcja document.write

Instrukcja `document.write()` pozwala na wyprowadzenie tekstu na ekran przeglądarki. Tekst, który chcemy wyświetlić, należy ująć w nawiasy i cudzysłowy i podać zaraz za `document.write()` np.

```
document.write ("Jaki miły mamy dzień!")
```

Ćwiczenie 2.2.

Napisz skrypt wyświetlający tekst „Jaki miły mamy dzień!" na ekranie przeglądarki.

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
</HEAD>
<SCRIPT language = "JavaScript">
document.write ("Jaki miły mamy dzień!")
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

```
document.write ("Jaki miły mamy dzień");
```

`document` to obiekt, który reprezentuje aktualną stronę. `write` to tzw. metoda, czyli pewna funkcja działająca na obiekcie `document` i, w tym przypadku, wyświetlająca na ekranie tekst. Tekst ten podajemy jako argument w nawiasach. Ogólnie można zapisać:

```
obiekt.metoda (argumenty metody)
```

Taki ciąg jest instrukcją i powinien zostać zakończony średnikiem. W JavaScript nie jest to jednak obligatoryjne, chyba że chcemy zapisać kilka instrukcji w jednej linii np.:

```
document.writeln ("Witamy");document.write ("na naszej stronie");
```

```
1  <HTML>
2  <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
3  <HEAD>
4  <PRE>
5  <SCRIPT>
6  document.writeln ("Witamy");document.write ("na naszej stronie");
7  </SCRIPT>
8  </PRE>
9  </HEAD>
10 <BODY>
11 </BODY>
12 </HTML>
13
```

Komentarze

Komentarz HTML

Znacznik `<SCRIPT>`, niezbędny do umieszczania kodu JavaScript, niestety nie jest częścią specyfikacji HTML 2.0, ani wcześniejszych, więc niektóre przeglądarki mogą go nie rozpoznać. W takiej sytuacji mogą one wyświetlić tekst skryptu na stronie. Chcielibyśmy oczywiście tego uniknąć.

Z pomocą przyjdą komentarze, które można umieszczać w kodzie HTML. Konstrukcja wygląda następująco:

```
<!--
Tekst komentarza
-->
```

Jeżeli zatem chcemy ukryć kod przed przeglądarkami nieobsługującymi JavaScript, powinniśmy ująć go w znaki komentarza, które są częścią standardu HTML.

Znacznik `<SCRIPT>`, niezbędny do umieszczania kodu JavaScript, niestety nie jest częścią specyfikacji HTML 2.0, ani wcześniejszych, więc niektóre przeglądarki mogą go nie rozpoznać. Co się stanie w takiej sytuacji? Otóż sam znacznik zostanie zignorowany, natomiast cały tekst skryptu znajdujący się między `<SCRIPT>` a `</SCRIPT>` *zostanie wyświetlony na ekranie, zmieniając nam treść i strukturę strony.* Chcielibyśmy oczywiście tego uniknąć. Z pomocą przyjdzie nam komentarz HTML, którego struktura wygląda następująco:

```
<!--
Tekst komentarza
-->
```

Jeżeli ujmiemy tekst skryptu w taką strukturę, przeglądarka nieobsługująca JavaScriptu pominie go, traktując właśnie jako zwykły komentarz.

Ćwiczenie 2.4.

Ukryj kod skryptu przed przeglądarkami nieobsługującymi JavaScript.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8 ">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
document.write ("Jaki miły mamy dzień!")
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Powyższe ćwiczenie obrazuje użycie komentarzy znanych z języka HTML. W JavaScript mamy natomiast dwie nowe możliwości zastosowania komentarza. Obie są zapożyczone z języków programowania takich C, C++ czy Java. Pierwszy typ komentarza składa się z dwóch ukośników: `//` (komentarz ten został zastosowany w poprzednim przykładzie, bowiem wczesne wersje przeglądarki Netscape Navigator nie rozpoznawały poprawnie sekwencji `-->` umieszczonej między etykietami `<SCRIPT>`). Zaczyna się on wtedy od miejsca wystąpienia tych dwóch znaków i obowiązuje do końca danego wiersza.

Komentarz typu //

Ćwiczenie 2.5.

Użyj komentarza składającego się z dwóch ukośników do opisania kodu skryptu.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
// Wyświetlenie napisu w oknie przeglądarki
document.write ("Hello, Jaki miły mamy dzień!")
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Komentarz blokowy

Komentarz może się również zaczynać od sekwencji `/*` i kończyć `*/`. W takim przypadku wszystko, co znajduje się pomiędzy tymi znakami, uznane zostanie za komentarz.

Ćwiczenie 2.6.

Użyj komentarza blokowego do opisania kodu skryptu.

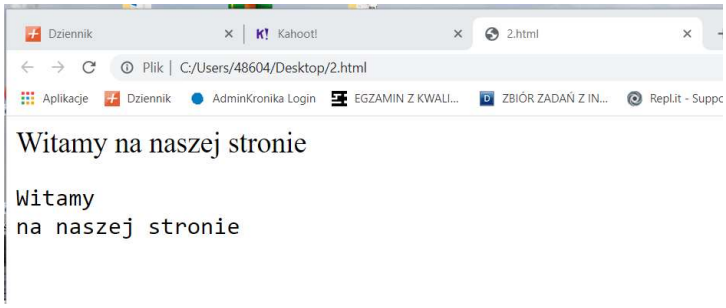
```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8 ">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nie osbługującymi JavaScript
/*
Komentarz blokowy
Wyświetlenie napisu w oknie przeglądarki
*/
document.write ("Hello, Jaki miły mamy dzień!")
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Formatowanie tekstu

Argumenty poznanych wyżej funkcji `write()` i `writeln()` są traktowane przez przeglądarkę jak tekst w HTML-u. Oznacza to, że możemy w łańcuchach wyświetlanych znaków wstawić praktycznie dowolne znaczniki formatujące tekst.

Ćwiczenie 2.9.

Użyj znaczników HTML formatujących tekst w argumentach funkcji `write()` i `writeln()`, tak by osiągnąć efekt jak na rysunku 2.4.



Rysunek 2.4. *Efekt użycia znaczników HTML w argumentach funkcji `write()` i `writeln()`*

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptów
document.write ("<FONT SIZE=+2>Witamy ");
document.write ("na naszej stronie");
document.writeln ("<PRE>Witamy");
document.write ("na naszej stronie</PRE></FONT>");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Oprócz znaczników HTML w wyświetlanych łańcuchach znakowych mogą też pojawić się znaki specjalne, takie jak np. rozpoczęcie nowego wiersza. Jeśli chcemy wyświetlić znak specjalny, musimy zastosować sekwencję — ukośnik (backslash) plus litera symbolizująca dany znak. Sekwencje te przedstawione są w tabeli 2.1.

Tabela 2.1. *Sekwencje znaków specjalnych*

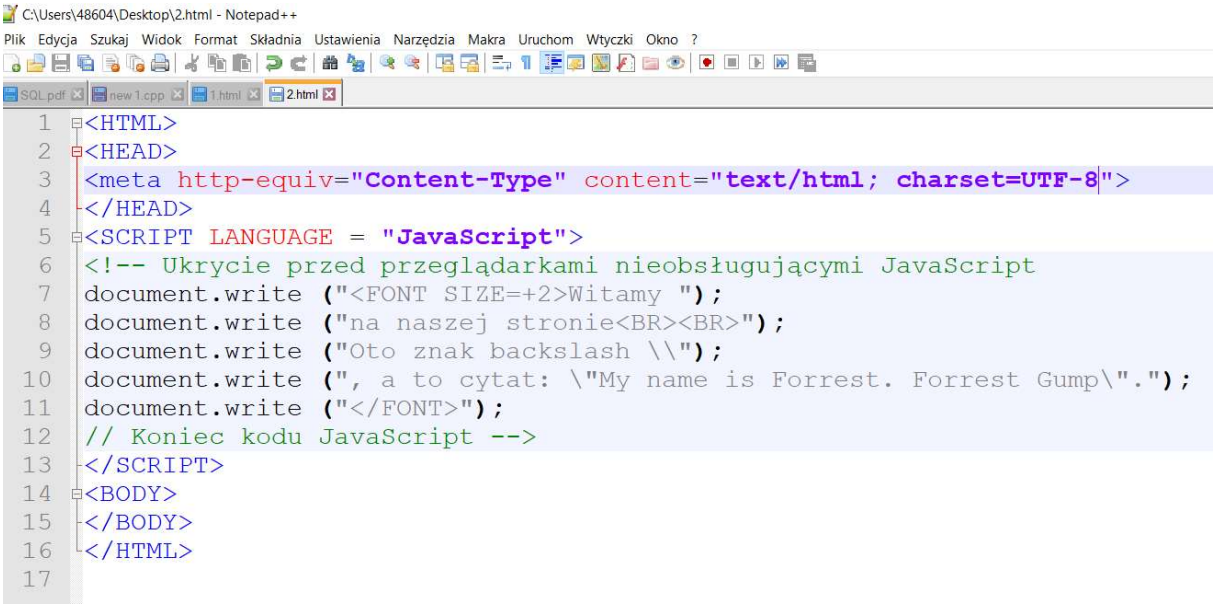
Sekwencja znaków specjalnych	Znaczenie
`\b`	Backspace
`\f`	wysunięcie kartki (ang. *form feed*)
`\n`	nowy wiersz (ang. *new line character*)
`\r`	enter (ang. *carriage return*)
`\t`	tabulator (ang. *tab character*)

Podobnie, jeżeli chcemy wyświetlić cudzysłów lub sam ukośnik (backslash \), musimy go poprzedzić znakiem backslash.

Ćwiczenie 2.10.

Używając funkcji `write()` wyprowadź na ekran tekst zawierający znak cudzysłowu oraz ukośnik (rysunek 2.5).

Rysunek 2.5. *Wyprowadzenie na ekran znaków specjalnych*



Okno dialogowe

Nauczymy się teraz, jak wyświetlić na ekranie najprostsze okienko dialogowe. Okno takie służy zwykle do poinformowania użytkownika o wystąpieniu jakiegoś zdarzenia. Najczęściej chodzi o sytuacje, w której wystąpił błąd. Na taki charakter prezentowanej metody wskazuje już sama nazwa: `alert()`. Może ona przyjmować jako parametr ciąg znaków, który zostanie wyświetlony na ekranie.

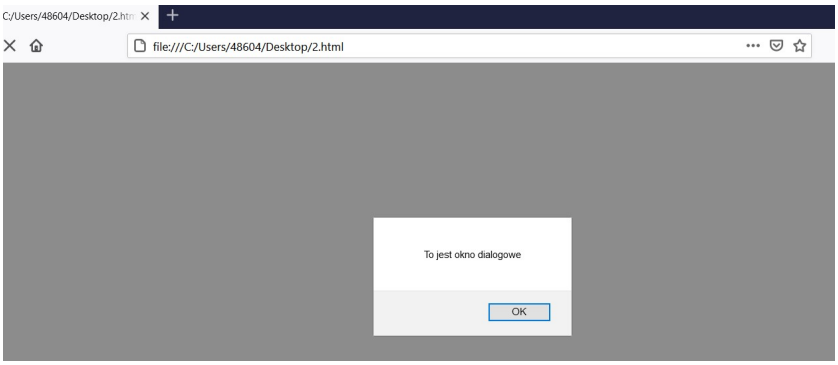
Ćwiczenie 2.12.

Wyświetl na ekranie okno dialogowe z dowolnym napisem.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
alert("To jest okno dialogowe");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Nasze okno wygląda jak na rysunku 2.7. Wykonywanie kodu jest wstrzymane do czasu, kiedy użytkownik kliknie przycisk *OK*. Dokładniej rzecz biorąc, w taki sposób powinna się zachować większość współczesnych przeglądarek. Tekst wyświetlany w oknie dialogowym możemy formatować, używając do tego celu znaków specjalnych (tabela 2.1), podobnie jak w przypadku funkcji `write()`.

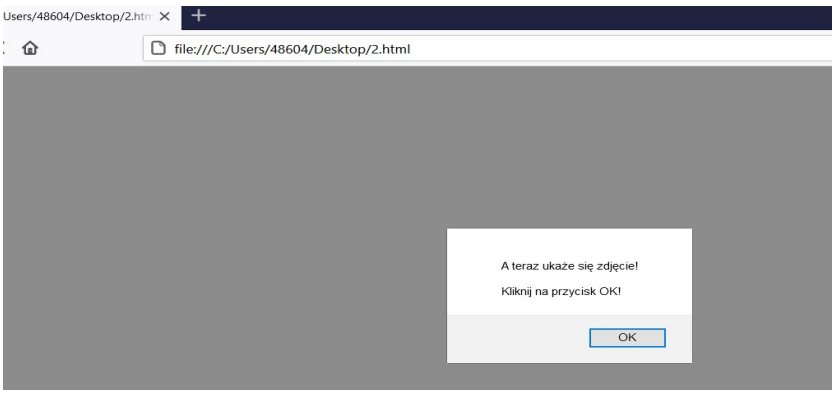
Rysunek 2.7.
Użycie funkcji `alert()` *do wyświetlenia okna dialogowego*



Ćwiczenie 2.13.

Wyświetl na ekranie okno dialogowe z tekstem w dwóch wierszach (jak na rysunku 2.8).

Rysunek 2.8.
Użycie znaków specjalnych formatujących tekst w oknie dialogowym




```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
alert ("\nA teraz ukaże się zdjęcie!\n\nKliknij na przycisk OK!")
document.write ("<IMG SRC = \"Przechwytywanie14.png\">");

// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Elementy języka JavaScript

Typy danych JavaScript

Do dyspozycji mamy cztery typy danych:

- liczby,
- wartości logiczne,
- łańcuchy znaków,
- wartość `NULL`.

Typ liczbowy

Służy do reprezentowania wszelkiego rodzaju liczb. Odmiennie niż w innych językach programowania, jak np. C++, gdzie do reprezentacji liczb służy co najmniej kilka typów danych, tutaj używamy tylko jednego — liczbowego. Liczby całkowite — zarówno dodatnie, jak i ujemne — możemy przedstawić w postaci dziesiętnej, szesnastkowej lub ósemkowej. System dziesiętny jest nam wszystkim znany; używamy w nim dziesięciu cyfr. Przypomnijmy jednak pokrótce podstawy dwóch pozostałych systemów.

Wartości logiczne

Zmienne tego typu mogą przyjmować tylko dwie wartości: `TRUE` i `FALSE` (prawda i fałsz). Będą one używane przy konstruowaniu wyrażeń logicznych, porównywania danych, wskazania, czy dana operacja zakończyła się sukcesem. Dla osób znających C czy C++ uwaga: wartości `TRUE` i `FALSE` nie mają przełożenia na wartości liczbowe, jak w przypadku wymienionych języków.

Łańcuchy znaków

Są to oczywiście dowolne ciągi znaków zawartych pomiędzy znakami cudzysłowów lub apostrofów. Mogą zawierać znaki specjalne. Przykładami mogą być: `"Kot ma Alę"`, `"liczby pierwsze: 1 3 5... "`.

Wartość NULL

Jest to pewien specjalny typ danych, który oznacza po prostu nic (`null`). Wartość ta jest zwracana przez niektóre funkcje. Bliżej zapoznamy się z nią w dalszej części książki.

Zmienne

Zmienne są to konstrukcje programistyczne, które pozwalają nam przechowywać dane. Ponieważ jednak JavaScript jest stosunkowo prostym językiem skryptowym, nie ma narzuconych takich ograniczeń. Zmiennych bowiem nie musimy (aczkolwiek możemy) deklarować przed użyciem; każda zmienna może też przyjmować

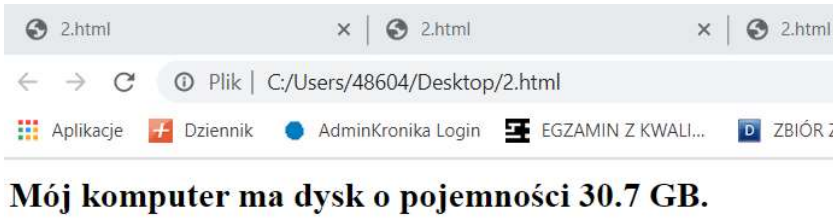
dane z dowolnego typu opisanego wyżej. Co więcej, typ danych przypisywanych zmiennej może się również zmieniać.

Ćwiczenie 3.1.

Zadeklaruj dwie zmienne, przypisz im dowolne ciągi znaków i wyprowadź je na ekran za pomocą funkcji `write()`.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
var zmienna1 = "Mój komputer";
var zmienna2 = 30.7;
document.write ("<H3>" + zmienna1 + " ma dysk o pojemności " + zmienna2 + "
GB.</H3>");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Zmiennej `zmienna1` przypisaliśmy ciąg znaków „`Mój komputer`”, zmiennej `zmienna2` natomiast wartość liczbową, dodatnią liczbę zmiennoprzecinkową `30.7`. Zmiennych tych użyliśmy jako argumentów funkcji `write()`. Musieliśmy również tak połączyć poszczególne łańcuchy tekstowe, aby otrzymać jeden, który ukazał się na ekranie. Do tego celu użyliśmy operatora + (plus). Nazywa się to łączeniem lub bardziej fachowo konkatenacją łańcuchów znakowych.



Rysunek 3.1. *Wyprowadzenie na ekran wartości dwóch zmiennych*

Wprowadzanie danych

okienko typu prompt()

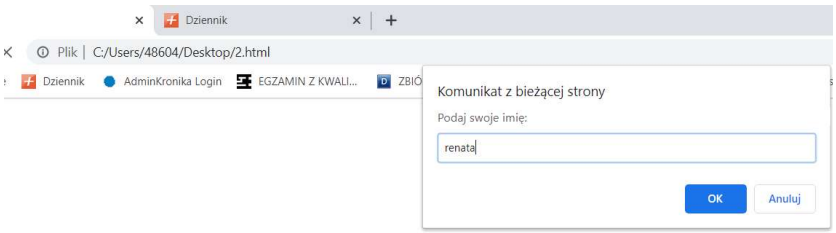
prompt('treść komunikatu', 'domyślna wartość');

Do metody prompt przekazujemy dwa parametry - jeden jest treścią, która będzie wyświetlana w okienku, a drugi jest domyślną wartością, która będzie wyświetlana w polu, w które wpisujemy tekst. Okienko to wyświetla dwa guziki: [OK] i [ANULUJ] (CANCEL). Jeżeli użytkownik kliknie [OK], to zostanie zwrócona wartość z pola tekstowego znajdującego się w tym okienku (lub też 'domyślna wartość', jeżeli użytkownik nie zmieniał zawartości tego pola). Jeżeli użytkownik kliknie [ANULUJ] (CANCEL), to zostanie zwrócona wartość null.

Ćwiczenie 3.4.

Wyświetl na ekranie okno dialogowe pozwalające na podanie przez użytkownika imienia. Następnie wyprowadź na ekran napis powitalny zawierający podane imię.

Rysunek 3.3. *Efekt działania funkcji `prompt()` w chrom*




```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
var imie = prompt ("Podaj swoje imię:");
document.write ("Cześć " + imie + "!");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

LUB

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
document.write ("Cześć " + prompt ("Podaj swoje imię:") + "!");
// Koniec kodu JavaScript -->
```

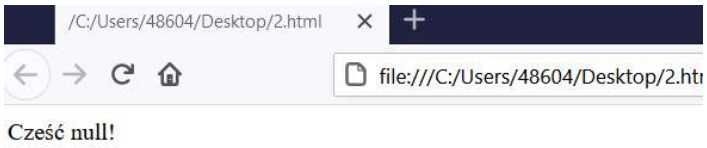
```
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Zadeklarowaliśmy zmienną o nazwie `imie`, której przypisaliśmy wartość zwracaną przez funkcję `prompt()`. Następnie wypisaliśmy wartość tej zmiennej, razem z tekstem powitania na ekran. Nie musimy jednak wcale deklarować zmiennej, aby uzyskać taki sam efekt. Wystarczy, że wywołanie funkcji `prompt()` umieścimy w argumencie funkcji `write()`.

Instrukcje warunkowe

Funkcja `prompt()` zwraca wartość podaną przez użytkownika lub jeśli nie podał on żadnej, wartość domyślną, tylko w przypadku naciśnięcia przycisku *OK*. Jeżeli jednak użytkownik wcisnął przycisk *Anuluj* (*Cancel*), zwrócona zostanie wartość `null`. W taki przypadku na ekranie ukaże się napis `Cześć null!`. Widać to na rysunku 3.5. Tego jednak chcielibyśmy uniknąć. Pomogą nam w tym instrukcje warunkowe.

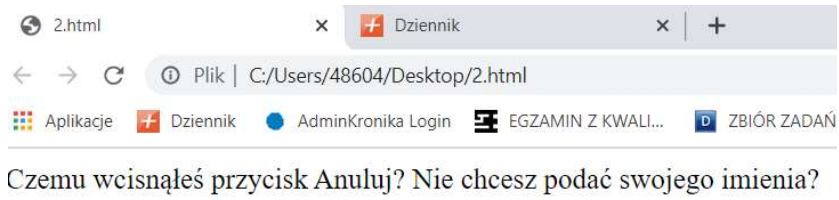
Rysunek 3.5. *Efekt wciśnięcia przycisku Anuluj (Cancel)*



Ćwiczenie 3.7.

Wyświetl na ekranie okno dialogowe pozwalające na podanie przez użytkownika imienia. Wyprowadź na ekran napis powitalny zawierający podane imię. W przypadku gdy użytkownik naciśnie przycisk *Anuluj* (*Cancel*), ma pojawić się stosowny komunikat (rysunek 3.6).

Rysunek 3.6. *Komunikat pojawia się po wciśnięciu przycisku Anuluj (Cancel) po wywołaniu funkcji `prompt()`*




```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
var imie = prompt ("Podaj swoje imię:", "");
if (imie == null) {
    document.write ("Czemu wcisnąłeś przycisk Anuluj? Nie chcesz podać swojego
imienia?");
}
else {
    document.write ("Cześć " + imie + "!");
}
// Koniec kodu JavaScript -->
</SCRIPT>
```

```
<BODY>
</BODY>
```

Ogólna konstrukcja `if...else` wygląda następująco:

```
if (warunek logiczny) {
    instrukcje do wykonania, jeśli warunek jest prawdziwy
}
else {
    instrukcje do wykonania, jeśli warunek nie jest prawdziwy
}
```