Obiekt document służy do reprezentacji wczytanego do przeglądarki dokumentu HTML oraz zawiera szereg właściwości i metod pozwalających na jego modyfikację. Poprzez ten obiekt można otrzymać dostęp praktycznie do każdego elementu strony i za jego pomocą można tymi elementami manipulować. Poniżej zostały zaprezentowane wybrane właściwości i metody.

Właściwości

Najczęściej wykorzystywane właściwości obiektu document zostały przedstawione w tabeli 4.4. Część z nich istnieje tylko ze względu na kompatybilność ze starszymi wersjami przeglądarek. Należy raczej unikać korzystania z takich właściwości jak: bgColor, fgColor, aLink, alinkColor, vLink, gdyż ich użyteczność jest znikoma.

Tabela 4.4. Wybrane właściwości obiektu document

Nazwa	Znaczenie	Dostępność
all	Obiekt zawierający odniesienia do wszystkich elementów dokumentu, charakterystyczny dla przeglądarki Internet Explorer	IE, OP
alinkColor	Kolor aktywnego odnośnika.	FF, IE, NN, OP
anchors	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Anchor.	FF, IE, NN, OP
applets	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Applet.	FF, IE, NN, OP
bgColor	Kolor tła dokumentu.	FF, IE, NN

body

Obiekt zawierający treść dokumentu HTML.

FF, IE, NN, OP

characterSet

Ciąg określający kodowanie znaków w dokumencie.

FF, NN

compatMode

Ciąg określający tryb kompatybilności dokumentu ze standardami HTML.

FF, IE, NN, OP

cookie

Ciąg znaków zawierający cookies danego dokumentu.

FF, IE, NN, OP

docType

Obiekt określający typ dokumentu (DTD).

FF, NN, OP

domain

Nazwa domenowa serwera, z którego pochodzi dokument.

FF, IE, NN, OP

embeds

Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów zagnieżdżonych.

FF, IE, NN, OP

fgColor

Kolor tekstu dokumentu.

FF, IE, NN, OP

fileCreatedDate

Data utworzenia pliku zawierającego aktualnie otwarty dokument, w formacie mm/dd/rrrr. Właściwość charakterystyczna

IE

fileModifiedDate

Data ostatniej modyfikacji pliku zawierającego aktualnie otwarty dokument, w formacie mm/dd/rrrr. Właściwość charakter

IE

fileSize

Rozmiar pliku zawierającego aktualnie otwarty dokument. Właściwość charakterystyczna dla przeglądarek z rodziny Intern

IE

forms

Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów formularzy.

FF, IE, NN, OP

height

Wysokość dokumentu.

FF, NN

images

Tablica zawierająca odniesienia do znajdujących się w dokumencie obrazów.

FF, IE, NN, OP

implementation

Obiekt typu DOMImplementation pozwalający stwierdzić, które elementy modelu DOM są implementowane przez bieżące

lastModified

Zwiera datę i czas ostatniej modyfikacji dokumentu.

FF, IE, NN, OP

linkColor

Definiuje kolor odnośnika.

FF, IE, NN, OP

links

Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Link (odnośników).

FF, IE, NN, OP

location

Obiekt przechowujący URL bieżącego dokumentu.

FF, IE, NN, OP

plugins

Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Plugin.

FF, IE, NN, OP

referrer

Zawiera URL dokumentu, z którego nastąpiło odwołanie do bieżącego dokumentu.

FF, IE, NN, OP

styleSheets

Zawiera wszystkie style zdefiniowane w dokumencie.

FF, IE, NN, OP

title

Zawiera tytuł dokumentu zdefiniowany za pomocą znacznika <title>.

FF, IE, NN, OP

URL

Ciąg zawierający URL bieżącego dokumentu.

FF, IE, NN, OP

vLink

Kolor odwiedzonego odnośnika.

FF, IE, NN, OP

width

Szerokość dokumentu.

FF, NN

Podstawowe informacje o dokumencie można więc uzyskać w wyniku zastosowania skryptu z listingu 4.13.

Listing 4.13. Wyświetlenie podstawowych informacji o dokumencie

```
<script type=&quot;text/javascript&quot;>
document.write(&quot;Podstawowe informacje o dokumencie:<br />&quot;);

document.write(&quot;Tryb kompatybilności: &quot;);
document.write(document.compatMode + &quot;<br />&quot;);

document.write(&quot;URL: &quot; + document.URL + &quot;<br />&quot;);

document.write(&quot;Liczba apletów: &quot;);
document.write(document.applets.length + &quot;<br />&quot;);

document.write(&quot;Liczba obrazów: &quot;);
document.write(document.images.length + &quot;<br />&quot;);

document.write(&quot;Liczba formularzy: &quot;);
document.write(document.forms.length + &quot;<br />&quot;);
```

document.write("Tytuł: " + document.title + "
");

document.write("Data ostatniej modyfikacji: ");

document.write(document.lastModified + "
");

</script>

Przykładowy efekt jego działania ilustruje rysunek 4.5.

Rysunek 4.5. Przykładowy efekt działania skryptu 4.13

Uzyskanie listy właściwości obiektu document dostępnych w danej przeglądarce można uzyskać za pomocą skryptu z listingu 4.14.

Listing 4.14. Lista właściwości obiektu document

<script type="text/javascript">

for(indeks in document){

  if(typeof document[indeks] != 'function'){

    document.write("document[" + indeks + "] = ");

    document.write(document[indeks]);

    document.write("
");

  }

}

</script>

Operator typeof został użyty, aby pominąć wyświetlanie metod obiektu document. Z metodą mamy do czynienia, gdy prawdziwy jest warunek document[indeks] == 'function'.

Metody

Obiekt document nie zawiera dużej liczby metod. Wybrane z nich zostały przedstawione poniżej. Najczęściej korzysta się z document.getElementById oraz document.write.

Metoda close

Wywołanie: document.close()

Dostępność: FF, IE3, NN2, OP9

Zamyka strumień wyjściowy otwarty za pomocą document.open. Przykład użycia znajduje się przy opisie metody open.

Metoda getElementById

Wywołanie: document.getElementById(id)

Dostępność: FF, IE5.5, NN6, OP9

Metoda getElementById zwraca odniesienie do elementu o identyfikatorze wskazywanym przez argument id. W ten sposób można uzyskać dostęp praktycznie do każdego elementu strony. Każdy taki element jest obiektem typu Element lub HTMLElement, którego najważniejszymi, z punktu widzenia twórcy strony, właściwościami są:

innerHTML — zawierająca kod HTML danego elementu;

style — określająca style danego elementu.

Należy przy tym zwrócić uwagę, że pierwsza z nich, choć często używana, nie jest częścią specyfikacji W3C i może być odmiennie implementowana w różnych przeglądarkach. Niemniej w większości przypadków zachowanie będzie bardzo podobne bądź identyczne.

Jeśli zatem za pomocą znacznika zostanie zdefiniowana warstwa HTML w postaci:

Akapit tekstowy

to zawartością innerHTML obiektu div o identyfikatorze wr1 będzie kod:

Akapit tekstowy

Zawartość tej właściwości może być zarówno odczytywana, jak i zapisywana. Oznacza to, że treść danego elementu strony może być dynamicznie modyfikowana, co pokazuje przykład widoczny na listingu 4.15.

Listing 4.15. Zmiana zawartości elementu strony

```
<!DOCTYPE HTML PUBLIC &amp;quot;-//W3C//DTD HTML 4.01//EN&amp;quot;
  &amp;quot;http://www.w3.org/TR/html4/strict.dtd&amp;quot;>
<html lang=&amp;quot;pl&amp;quot;>
 <head>
  <meta http-equiv=&amp;quot;Content-Type&amp;quot; content=&amp;quot;text/html; charset=utf-8&amp;quot;>
  <meta http-equiv=&amp;quot;Content-Script-Type&amp;quot; content=&amp;quot;text/javascript&amp;quot;>
  <title>Moja strona WWW</title>
 </head>
 <body>
  <div id=&amp;quot;wr1&amp;quot;>
```

```
    </div>
    <script  type=&quot;text/javascript&quot;>
      var div = document.getElementById(&quot;wr1&quot;);
      div.innerHTML =  &quot;<p>Akapit tekstowy</p>&quot;;
    </script>
  </body>
</html>
```

W kodzie strony znalazła się warstwa zdefiniowana za pomocą znacznika . Ma ona identyfikator wr1 (id="wr1"). Warstwa jest pusta (a dokładniej zawiera kilka znaków spacji i znak końca wiersza, ale nie będą one wyświetlone na witrynie). Taka strona powinna być więc pusta, jednak po jej wczytaniu do przeglądarki pojawi się napis Akapit tekstowy. Jest za to odpowiedzialny skrypt JavaScript znajdujący się za definicją warstwy. Pobiera on odwołanie do warstwy o identyfikatorze wr1 (document.getElementById("wr1")) i zapisuje je w zmiennej pomocniczej div. Następnie właściwości innerHTML, odpowiadającej kodowi HTML zapisanemu w danym elemencie strony, przypisuje ciąg znaków Akapit tekstowy . To oznacza, że wewnątrz warstwy znajdzie się definicja akapitu tekstowego. Dlatego też pojawia się on na ekranie. Użyta w akapicie sekwencja specjalna pozwala na zachowanie zgodności kodu źródłowego ze standardem HTML. Oczywiście sekwencja / jest zamieniana przez przeglądarkę na pojedynczy znak /.

Powyższy przykład, choć demonstruje tworzenie elementu strony przez skrypt, jest statyczny i bez zaglądania do kodu źródłowego trudno stwierdzić, że została użyta technika dynamicznego wstawiania elementu. Jeśli jednak użyjemy poznanej wcześniej metody setInterval, będziemy mogli zobaczyć dynamiczną zmianę zawartości elementu witryny. Tak działa przykład przedstawiony na listingu 4.16.

Listing 4.16. Dynamiczna zmiana zawartości elementu witryny

```
<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.01//EN&quot;
  &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>
<html lang=&quot;pl&quot;>
  <head>
    <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>
    <meta  http-equiv=&quot;Content-Script-Type&quot;  content=&quot;text/javascript&quot;>
    <title>Moja strona  WWW</title>
    <script  type=&quot;text/javascript&quot;>
      var i = 1;
      function changeContent()
      {
```

```
      var div =  document.getElementById(&quot;wr1&quot;);

      div.innerHTML =  &quot;<p>Akapit tekstowy nr &quot; + ++i + &quot;</p>&quot;;

    }

     setInterval(&quot;changeContent();&quot;, 1000);

   </script>


 </head>

 <body>

   <div id=&quot;wr1&quot;>

    <p>Akapit tekstowy nr 1</p>

   </div>

 </body>

</html>
```

Akapit tekstowy został umieszczony na warstwie, której został przypisany identyfikator wr1. Dzięki temu za pomocą wywołania document.getElementById("wr1") można się do niej odwoływać. Odwołanie to następuje w funkcji changeContent. Obiekt odpowiadający warstwie wr1 jest pobierany i przypisywany zmiennej div, po czym następuje wymiana treści HTML zapisanej w warstwie: zamiast istniejącej liczby jest zapisywana wartość zmiennej i. Ponieważ funkcja changeContent (dzięki wywołaniu setInterval("changeContent();", 1000);) jest wywoływana co sekundę, a zmienna i jest za każdym razem zwiększana o 1, na ekranie pojawią się kolejne numery akapitów.

Skoro wiadomo już, jak wykorzystać właściwość innerHTML, warto użyć również właściwości style. Ponieważ pozwala ona zarówno na odczytanie, jak i zapisanie atrybutów stylów CSS danego elementu strony, umożliwia bardzo szeroką modyfikację jego wyglądu i zachowania. Należy przy tym zwrócić uwagę, że jeśli dany trybut jest definiowany jako:

nazwa-atrybutu

to z reguły odwołanie do niego będzie miało postać:

nazwaAtrybutu

przykładowo atrybut:

font-weight

jako właściwość obiektu style będzie miał postać:

fontWeight

Na listingu 4.17 zobrazowano, jak wykorzystać go do modyfikacji wyglądu tekstu.

Listing 4.17. Dynamiczna modyfikacja stylu CSS

```
<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.01//EN&quot;
  &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>
<html lang=&quot;pl&quot;>
 <head>
  <meta http-equiv=&quot;Content-Type&quot; content=&quot;text/html; charset=utf-8&quot;>
  <meta http-equiv=&quot;Content-Script-Type&quot; content=&quot;text/javascript&quot;>
  <title>Moja strona WWW</title>
  <script type=&quot;text/javascript&quot;>
   var styl = &quot;normal&quot;;
   function changeStyle()
   {
    var element = document.getElementById(&quot;at1&quot;);
    if(styl == &quot;bold&quot;){
     styl = &quot;normal&quot;;
    }
    else{
     styl = &quot;bold&quot;;
    }
    element.style.fontWeight = styl;
   }
   setInterval(&quot;changeStyle();&quot;, 1000);
  </script>
 </head>
 <body>
  <p id=&quot;at1&quot; style=&quot;font-weight:normal&quot;>Akapit tekstowy nr 1</p>
 </body>
</html>
```

W treści strony tym razem znalazł się akapit tekstowy o identyfikatorze at1. Dzięki instrukcji setInterval("changeStyle();", 1000); co 1 sekundę jest wywoływana funkcja changeStyle, która dokonuje zmiany stylu akapitu. O tym, jaka wartość zostanie przypisana atrybutowi font-weight, czyli właściwości fontWeight obiektu style, decyduje globalna zmienna styl. Początkowo jej wartością jest normal. Każde wywołanie funkcji changeStyle powoduje zmianę wartości zapisanej w zmiennej styl na przeciwną (czyli z normal na bold lub z bold na normal). Dzięki temu co sekundę będzie następowała zmiana grubości tekstu. **Metoda getElementsByName**

Wywołanie: document.getElementsByName(name)

Dostępność: FF, IE5.5, NN6, OP9

Metoda getElementsByName pobiera listę elementów posiadających atrybut name o wartości wskazanej przez argument name (zwracany jest obiekt będący kolekcją). Dotyczy to oczywiście tylko takich elementów, które ten atrybut mogą posiadać. Przykładowo: jeśli w dokumencie zostaną zdefiniowane dwie warstwy o takiej samej nazwie i różnych identyfikatorach:

to wywołanie:

var el = document.getElementsByName("a1");

spowoduje przypisanie zmiennej el listy o dwóch elementach. Dostęp do poszczególnych elementów można uzyskać przez zastosowanie odwołania:

lista.item(indeks_elementu)

np.:

alert(el.item(0).id);

alert(el.item(1).id);

Możliwe jest również zastosowanie składni z nawiasem kwadratowym, schematycznie:

lista[indeks_elementu]

np.:

alert(el[0].id);

alert(el[1].id); **Metoda getElementsByTagName**

Wywołanie: document.getElementsByTagName(tag)

Dostępność: FF, IE5.5, NN6, OP9

Metoda getElementsByTagName pobiera listę elementów utworzonych za pomocą znacznika określonego przez argument tag. Jeśli zatem istnieje potrzeba uzyskania dostępu do wszystkich odnośników zawartych w danym dokumencie, to należy zastosować wywołanie:

document.getElementsByTagName("a");

a gdyby była potrzebna lista wszystkich warstw, wywołanie:

document.getElementsByTagName("div");

Uzyskamy w ten sposób obiekt zawierający wszystkie wskazane elementy. Dostęp do poszczególnych elementów odbywa się w taki sam sposób, jaki został opisany przy metodzie getElementsByName. **Metoda open**

Wywołanie: document.open([typ[, replace]])

Dostępność: FF, IE5.5, NN6, OP9

Metoda open otwiera strumień pozwalający na zapis dokumentu za pomocą metod write i writeln. Aktualne dane zawarte w dokumencie zostaną usunięte. Po zakończeniu wysyłania danych należy użyć metody close do zamknięcia strumienia. Parametr typ powinien określać typ danych. Jeżeli nie zostanie użyty, przyjęty zostanie typ domyślny text/html. Parametr replace pozwala określić, czy nowy dokument ma wystąpić jako nowy w historii odwiedzonych witryn (brak parametru replace), czy też zastąpić bieżący wpis w historii witryn (parametr replace równy true). Metod open i close można użyć następująco:

var dane = " akapit ";

document.open("text/html", true);

document.write(dane);

document.close();

W praktyce używanie open i close w typowych zastosowaniach nie jest konieczne, gdyż metody write i writeln same otwierają i zamykają strumień danych. **Metoda write**

Wywołanie: document.write(text)

Dostępność: FF, IE4, NN4, OP7

Metoda write umieszcza w dokumencie tekst przekazany w postaci argumentu text. Przykłady jej wykorzystania pojawiały się już wielokrotnie w trakcie kursu. **Metoda writeln**

Wywołanie: document.writeln(text)

Dostępność: FF, IE4, NN4, OP7

Metoda writeln działa analogicznie do write, z tą różnicą, że na końcu tekstu przekazanego w postaci argumentu text jest dodatkowo umieszczany znak końca linii.