Tworzenie Stron Internetowych

odcinek 11

JavaScript

JavaScript (ECMAScript) – skryptowy język programowania powszechnie używany w Internecie. Skrypty JS dodają do stron WWW interaktywność i funkcjonalności, np.: reagowanie na zdarzenia generowane przez użytkownika, sprawdzanie poprawności formularzy, budowanie elementów nawigacyjnych.

Początki JS sięgają 1995 roku (Brendan Eich, Netscape). Za standaryzację odpowiada Ecma International (European association for standardizing information and communication systems). Standard JS jest opisany w specyfikacji oznaczonej jako ECMA-262.

Skrypty JS nie wymagają kompilacji. Ich używanie jest darmowe.

Język JS jest standardem ISO.

Możliwe jest też pisanie w JS zwykłych aplikacji.

JavaScript to nie to samo co Java. To są dwa różne języki programowania.

JS – edytory

Edycja plików JS

Skrypt JS to plik tekstowy, można więc go pisać w dowolnym edytorze tekstowym. Warto jednak używając edytorów programistycznych, które ułatwiają pisanie kodu. Dobry edytor JS posiada:

- podświetlanie składni (unikanie błędów w poleceniach),
- zaawansowana edycja (wielopoziomowe cofanie, znajdź-zastąp,…),
- generatory elementów JS .

Często edytory do tworzenia stron internetowych obsługują edycję dokumentów HTML, CSS i JS.

JS – zalety

Dlaczego używać JS:

- **składnia JS jest prosta** – twórcy stron internetowych mogą wykorzystać funkcjonalność JS nie będąc programistami
- **JS zwiększa interakcję z użytkownikiem** – skrypty mogą reagować na zdarzenia generowane po stronie użytkownika
- **JS czyta i zapisuje elementy HTML** – skrypty mogą odczytać i zmienić zawartość dowolnego elementu HTML oraz je ukrywać, pokazywać, kasować, tworzyć i powielać. Mogą też zmieniać atrybuty elementów HTML oraz ich styl CSS.
- **JS może sprawdzać poprawność danych** – skrypty pozwalają sprawdzić dane wpisane w formularz przed ich wysłaniem, co oszczędza pracy serwerowi
- **JS wykryje z jakiej przeglądarki korzysta użytkownik** – skrypt wykrywając typ przeglądarki może załadować wersję strony zaprojektowaną dla niej
- **JS może tworzyć ciasteczka (cookies)** – skrypt pozwala przetrzymywać i odczytywać informacje na komputerze użytkownika
- **skrypty JS możemy pisać sami lub wykorzystać gotowe**, pobrane z sieci (np.: www.kurshtml.edu.pl/skrypty/)

JS – osadzanie

Skrypt JS umieszczane są w dokumencie HTML (<head> lub/i <body>) za pomocą znacznika **<script>** w postaci:

- samego kodu JS

```
<script type="text/javascript">
/* <![CDATA[ */
 kod JS
/*]]> */</script>
```

obecnie atrybut type nie jest wymagany

zabezpiecza przed interpretacją kodu jako składni HTML, tylko dla xHTML

- odnośnika do pliku zawierającego kod JS

```
<script type="text/javascript" src="plik.js"></script>
```

Dodatkowa zaleca się w części **<head>** dokumentu HTML deklarację typu użytego języka skryptowego:

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

Kolejność osadzania skryptów ma znaczenie, ponieważ wykonywane są one w takiej kolejności, w jakiej zostały umieszczone w kodzie strony.

Jeśli przeglądarka użytkownika nie obsługuje JS, to nie będzie miał on dostępu do treści/funkcji generowanych przez skrypt JS. O tej sytuacji możemy użytkownika poinformować używając znacznika **<noscript>**:

```
<noscript>Uwaga, Twoja przeglądarka nie obsługuje JavaScript!</noscript>
```

JS – osadzanie

Zalety umieszczania skryptów JS osobnych plikach:

- ten sam kod można wykorzystać do wielu stron WWW
- oddzielenie kodu HTML i JS
- ułatwienie edycji kodów
- szybsze ładowanie stron WWW

Jest też możliwość użycia zewnętrznych skryptów przez podanie ich adresu URL

```
<script src="https://www.jakas-strona.com/js/skryptJS-1.js"></script>
```

JS – składnia

Komentarz

Komentarze w kodzie JS umieszczany są następująco:

- blokowo – pozwala wykomentować wiele linii

```
/* komentarz blokowy
zajmuje kilka linii */
```

- liniowo – obowiązuje tylko do końca jednej linii

```
// komentarz liniowy
```

Ogólnie o składni

Kod (skrypt) JS składa się z zestawu instrukcji. **Poszczególne instrukcje oddzielane są od siebie średnikiem.**

Przeglądarka wykonuje instrukcje JS w kolejności ich umieszczenia w kodzie.

Umieszczając zestaw instrukcji w nawiasach { } tworzymy blok kodu, który ma być wykonywany jako całość.

Dla lepszej czytelności kodu dobrze jest unikać długich linii oraz używać spacji (np. przy operatorach).

JS – składnia

Dane

Dostępne są następujące typy danych: **typ liczbowy** (całkowity i rzeczywisty), **typ łańcuchowy**, **typ logiczny**, **typ obiektowy** i **typy specjalne**.

Zmienne

Zmienne definiujemy poprzez przypisaniu wartości lub instrukcją **var**. Nazwa zmiennej zaczyna się od litery lub znaku podkreślenia i może zawierać dodatkowo liczby. Duże i małe litery są rozróżniane. Zmienne zdefiniowane poza funkcjami są globalne, czyli widoczne są dla całego skryptu, wewnątrz funkcji – lokalne, jeśli są zdefiniowane przez **var**.

Przykłady:

a=100; var *a;*

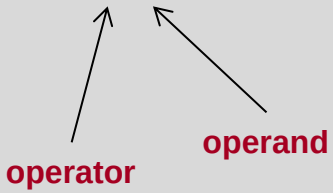
a=b; var *c* = 50*;*

Na zmiennych można wykonywać następujące operacje: arytmetyczne, przypisania, porównywania, bitowe, logiczne i inne.

Więcej informacji w dokumentacji, np. *webmaster.helion.pl/index.php/kjs-cechy-jezyka/kjs-operatory*

Przykłady:

4 + 7 var *suma* = 2 + 3 + 4;



JS – składnia

Instrukcje sterujące

JavaScript umożliwia wykorzystanie standardowych instrukcji sterujących takich jak:

- instrukcje warunkowe **if**, **if…else**, **if…else if**
- instrukcje wyboru: **switch…case**
- operator warunkowy: **?**
- pętle: **for**, **for...in**, **while**, **do...while**

Więcej informacji o instrukcjach w dokumentacji JS, np.

webmaster.helion.pl/index.php/kjs-cechy-jezyka/kjs-instrukcje

JS – składnia

Funkcje

W JavaScript możemy tworzyć własne funkcje (wydzielone bloki kodu przeznaczone do wykonywania konkretnych zadań) w następujący sposób:

```
function nazwa_funkcji(argument1, argument2, …)
{
instrukcje wnętrza funkcji
}
```

przykład (dodawanie dwóch liczb):

```
<script>
   function dodaj(var1, var2)
   {
      var wynik = var1 + var2;
      return wynik;

   }
   var suma = dodaj (1, 2);
   document.write("Wynikiem dodawania 1 + 2 jest " + suma + ".");



</script>
```

przerwanie działania funkcji i zwrot wyniku

wywołanie funkcji z określonymi argumentami

instrukcja pozwalająca na umieszczanie tekstu na stronie www

JS – składnia

Tablice

Jak w innych językach, dostępne są też w JS tablice pozwalające przechowywać zbiory elementów. Komórki tablicy numerowane są od zera. Tablice deklarowane są następująco:

```
var nazwa_tablicy = Array();
var nazwa_tablicy = Array(rozmiar_tablicy);
var nazwa_tablicy = new Array("wartość-1", "wartość-2", …, "wartość-n");
var nazwa_tablicy = [element-1, element-2, ..., element-n];
var nazwa_tablicy = [];
```

Podanie rozmiaru nie jest konieczne, będzie on dynamiczne zmieniany wraz ze wstawianiem kolejnych wartości. Do elementu tablicy odwołujemy się przez indeks:

```
nazwa_tablicy[indeks];
```

Do każdego elementu tablicy można przypisać wartość:

```
nazwa_tablicy[indeks] = wartość;
```

lub ją usunąć:

```
nazwa_tablicy[indeks] = null;
nazwa_tablicy[indeks] = "";
```

Możemy też usunąć komórkę tablicy:

```
delete nazwa_tablicy[indeks]
```

JS – składnia

Obiekty, funkcje globalne

W języku JavaScript dostępne są **predefiniowane obiekty** oraz **funkcje globalne**. Obiekty rozszerzają możliwości języka i (czasami) usprawniają pracę. Obiekt przechowuje dane (właściwości obiektu) wraz z operacjami, które można na nich wykonać (metody obiektu). Dostęp do właściwości lub metody możliwy jest następująco:

nazwa_obiektu.nazwa_właściwości

nazwa_obiektu.nazwa_metody(*argumenty metody*)

Dostępne funkcje globalne (wybrane)

- Funkcja **eval** (eval(*str*)) – zwraca wartość wyrażenia *str* lub wykonuje instrukcję *str*.
- Funkcja **isNaN** (isNaN(*wartość*)) – zwraca wartość *false*, jeżeli parametr *wartość* jest liczbą lub *true* w przeciwnym razie.
- Funkcja **parseInt** (parseInt(*str[, podstawa]*) – przetwarza ciąg znaków podany argumentem *str* na wartość całkowitą. Opcjonalny argument *podstawa* pozwala na ustalenie podstawy systemu liczbowego.
- Funkcja **parseFloat** (parseFloat(*str*)) - przetwarza ciąg znaków podany argumentem *str* na wartość rzeczywistą. Jeżeli argument nie przedstawia prawidłowej wartości rzeczywistej, funkcja zwróci wartość NaN.
- Funkcja **isFinite** (isFinite(*wartość*)) - zwraca wartość *true*, jeżeli parametr *wartość* ma wartość skończoną lub *false* w przeciwnym razie.

JS – składnia

Obiekty, funkcje globalne

W języku JavaScript dostępne są **predefiniowane obiekty** oraz **funkcje globalne**. Obiekty rozszerzają możliwości języka i (czasami) usprawniają pracę. Obiekt przechowuje dane (właściwości obiektu) wraz z operacjami, które można na nich wykonać (metody obiektu). Dostęp do właściwości lub metody możliwy jest następująco:

nazwa_obiektu.nazwa_właściwości

nazwa_obiektu.nazwa_metody(*argumenty metody*)

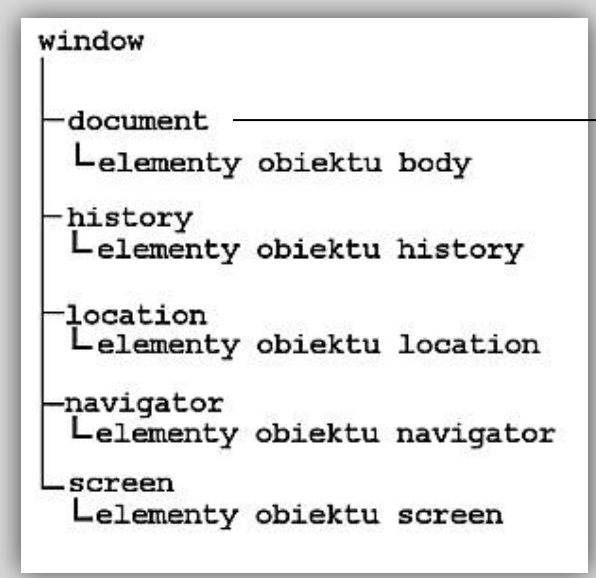
Dostępne predefiniowane obiekty

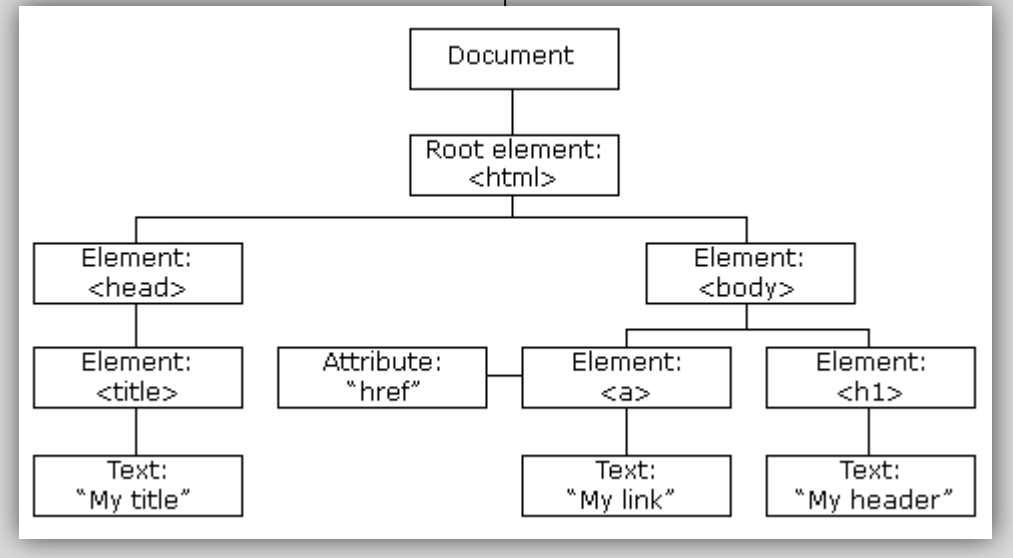
- Obiekt **Array**
- Obiekt **String**
- Obiekt **Date**
- Obiekt **Math**
- Obiekt **Number**
- Obiekt **Boolean**
- Obiekt **RegExp**

JS – DOM

Document Object Model (obiektowy model dokumentu)

Skryptami JS można sterować elementami strony lub zachowaniem przeglądarki. Takie działanie możliwe jest dzięki **obiektowemu modelowi dokumentu** (**DOM**). W modelu tym przeglądarka wraz z wyświetlana treścią jest zestawem obiektów. Obiekty w DOM ułożone są hierarchicznie (patrz obrazek).





JS – DOM

Document Object Model (obiektowy model dokumentu)

DOM jest interfejsem pozwalającym na dynamiczny dostęp i wprowadzanie zmian w zawartości, strukturze i stylu dokumentu. Dzięki temu, z poziomu JS, elementy składające się na drzewo dokumentu HTML stają się obiektami na których możemy wykonywać szereg działań.

Wykorzystując DOM, JavaScript może:

- zmienić dowolny istniejący element HTML
- zmienić dowolny istniejący atrybut
- zmienić dowolną istniejącą deklarację CSS
- usunąć dowolny istniejący element HTML i atrybut
- dodać nowe elementy HTML lub atrybuty
- odpowiadać na dowolne zdarzenie HTML obecne na stronie
- tworzyć nowe zdarzenia HTML

W DOM elementy HTML są obiektami, których cechy, metody dostępu i zdarzenia są zdefiniowane.

JS – DOM

Document Object Model

Obiekty DOM

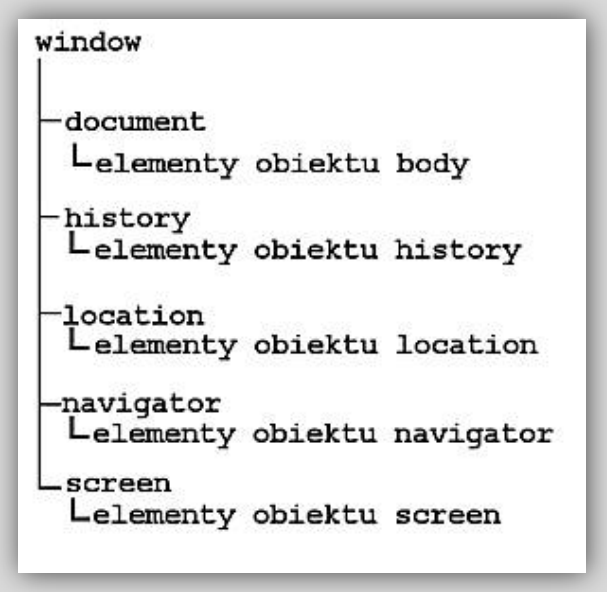
- **window** – znajduje się na szczycie hierarchii i przedstawia okno przeglądarki. Jest obiektem domyślny i przy wywoływaniu jego metod i własności można stosować zapis skrócony, np.:

  alert("*tekst*"); zamiast window.alert("*tekst*"); wyświetlanie okna dialogowego z *tekstem*

  lista metod i własności obiektu window dostępna jest np. tu

  webmaster.helion.pl/index.php/kursjs-wspolpraca-z-przegladarka-model-dom/kursjs-obiekt-window

  Możliwe są m.in. następujące operacje: otwieranie okna, zmiana jego rozmiarów i położenia, wyświetlanie okien dialogowych, paska przewijania, …



JS – DOM

Document Object Model

Obiekty DOM, cd:

- **document** – umożliwia sterowanie wyświetlanym dokumentem HTML; daje dostęp do każdego elementu tego dokumentu pozwalając na jego zmianę. Obiekt ten posiada sporo właściwości (*np.: URL dokumentu, tytuł, odniesienia do obrazów, data ostatniej modyfikacji,…*). Najważniejsze (najczęściej stosowane) metody obiektu document to:
  - getElementById(*id*) – zwraca informacje o elemencie HTML posiadającym zadane id (kod elementu i jego styl lokalny). Umożliwia dostęp do różnych części kodu i np. zmianę ich zawartości.

    Przykład:

```
<body>

  <div id="et1">

  </div>

  <script>
    var div_var = document.getElementById("et1");
    div_var.innerHTML = "<p>jakiś tekst</p>";
  </script>

</body>
```

wpisze do div o podanym id kod akapitu

JS – DOM

Document Object Model

Obiekty DOM, cd:

- **document** – umożliwia sterowanie wyświetlanym dokumentem HTML; daje dostęp do każdego elementu tego dokumentu pozwalając na jego zmianę. Obiekt ten posiada sporo właściwości (*np.: URL dokumentu, tytuł, odniesienia do obrazów, data ostatniej modyfikacji,…*). Najważniejsze (najczęściej stosowane) metody obiektu document to:
  - write(*tekst*) oraz writeln(*tekst*) – umieszcza w dokumencie zadany *tekst*. Druga metoda dodaje na końcu tekstu znak końca linii.

    Przykład:

```
<script>
  document.write("<p  id="akapit">");
  document.write("To jest treść napisana " + 22 + " maja o godzinie " + var);
  document.write("</p>");
</script>
```

JS – DOM

Document Object Model

Obiekty DOM, cd:

- **history** – zawiera historię odwiedzin stron dokonanych przez użytkownika podczas danej sesji przeglądarki. Posiada zaledwie kilka metod i właściwości. Wśród nich:
  - current – adres URL aktualnego dokumentu,
  - previous, next – adresy strony poprzedniej i następnej
  - back, forward – wczytuje poprzedni, następny dokument
  - go(*parametr*) – wczytuje dokument wskazany przez parametr. Liczba całkowita oznacza pozycję na liście historii odwiedzin (bieżący = 0, wstecz z minusem). Można też zadać parametr jako adres URL z listy historii
- **location** – podaje szczegółowe informacje o adresie URL bieżącego dokumentu. Właściwości tego obiektu podają pełny adres oraz jego części składowe, np. nazwę protokołu (https, http, ftp). Przykładowe metody:
  - assign(*url*) – wczytuje dokument o adresie wskazanym przez argument *url*.
  - reload(*argument*) – wymusza ponowne wczytanie bieżącej strony z serwera jeśli *argumet* ma wartość *true* lub z pamięci przeglądarki w przeciwnym wypadku.
  - replace(*url*) – zastępuje bieżący dokument przez wczytany spod adresu wskazanego przez argument *url*. Nadaje się do wykorzystanie w przypadku, gdy przenosimy stronę pod inny adres.

JS – DOM

Document Object Model

Obiekty DOM, cd:

- **navigator** – zawiera następujące informacje: nazwa wersja i język przeglądarki, typ i język systemu operacyjnego, obsługa plików cookie, zainstalowane rozszerzenia, … Metodą tego obiektu jest javaEnabled(), która zwraca wartość *true*, jeśli przeglądarka obsługuje Javę lub *false* w przeciwnym razie.

Więcej na temat właściwości i metod obiektów modelu DOM można znaleźć np. tu:

webmaster.helion.pl/index.php/kursjs-wspolpraca-z-przegladarka-model-dom

JS – obsługa zdarzeń

Skrypty JS mogą obsługiwać zdarzenia, czyli wpływać na wyświetlanie treści dokumentu w odpowiedzi na akcję wykonaną przez użytkownika lub przeglądarkę.

Procedurę obsługującą zdarzenia wprowadzamy do kodu HTML w postaci atrybutów:

```
<element zdarzenie="instrukcja">
<element zdarzenie="nazwa_funkcji()">
```

Poprawne jest użycie ” ” lub ’ ’.

Przykład:

```
<button onclick="document.getElementById('demo').innerHTML=Date()">podaj datę i godz.</button>

<p id="demo"></p>
```

po kliknięciu przycisku skrypt wypisze aktualną datę i godzinę

Więcej na temat obsługi zdarzeń można znaleźć np. tu:

http://webmaster.helion.pl/index.php/kursjs-obsluga-zdarzen-i-elementow-strony/kursjs-zdarzenia

JS – automat do zapisywania LM

Poniżej zamieszczam kod umożliwiający automatyczne wpisanie jasności LM po podaniu liczby gwiazd N.

```
<td>                                                    komórka tabeli, w której użytkownik wybiera odpowiednią liczbę N
  <select name="N-pole3" id="pole3" size="1">
    <option value="5">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
    <option value="9">9</option>
    <option value="11">11</option>
    <option value="13">13</option>
    <option value="14">14</option>
    <option value="15">15</option>
    <option value="16">16</option>
    <option value="17">17</option>
    <option value="18">18</option>
    <option value="19">19</option>
    <option value="20">20</option>
    <option value="23">23</option>
    <option value="25">25</option>
    <option value="27">27</option>
    <option value="29">29</option>
    <option value="33">33</option>
    <option value="37">37</option>
    <option value="44">44</option>
    <option value="49">49</option>
    <option value="54">54</option>
  </select>
  <button onclick="NdoLMpole3()">OK</button>        komórka zawiera też przycisk, którym użytkownik zatwierdza wybór, co powoduje wywołanie funkcji napisanej w JS
 </td>

 <td id="wynpole3"> </td>                           komórka, do której ma być wpisana wartość LM – w efekcie działania skryptu JS
```

Ciąg dalszy na następnej stronie.

JS – automat do zapisywania LM

Blok <script> z funkcją wywoływaną po kliknięciu przycisku w tabeli.

```
<script>
   function NdoLMpole3() {
   var inpN = document.getElementById("pole3").value;
   var Npole3 = ["5", "6", "7", "8", "9","11","13","14","15","16","17","18","19","20","23","25","27","29","33","37","44","49","54"];
   var LMpole3 = ["4.5","4.6","4.8","5.2","5.4","5.7","5.8","6.0","6.1","6.2","6.3","6.4","6.5","6.6","6.7","6.8","6.9","7.0","7.1","7.2","7.3","7.4","7.5"];
   var indN = Npole3.indexOf(inpN);
   document.getElementById("wynpole3").innerHTML = LMpole3[indN];
   }
</script>
```

funkcja pobiera wartość N wybraną przez użytkownika a następnie znajduje odpowiadajacą jej wartość LM i wpisuje ją w odpowiedniej komórce w tabeli

Podobny kod należy napisać dla pozostałych obszarów nieba znajdujących się w raporcie.