

JavaScript - wykład 2 Document Object Model

Beata Pańczyk

Plan wykładu

- Co to jest DOM?
- DOM W3C
- DOM HTML
- Metody DOM
- Wyszukiwanie elementów
- Przykłady

2

DOM W3C

- Służy do przedstawienia dokumentów HTML i XML (XHTML)
- CSS i JavaScript używają go do operowania na dokumentach
- Wg W3C DOM: dokument to drzewiasty zbiór węzłów
- Węzeł może być elementem, ciągiem tekstu, komentarzem, itp.
- Każdy węzeł może mieć jednego rodzica, dowolną ilość braci (lub sąsiadów) i jeśli jest elementem, dowolną ilość dzieci.

3

DOM W3C



4

Metody DOM

Węzeł	Selektor CSS	JavaScript DOM
Korzeń (Root)	:root, html	element.ownerDocument.documentElement
Rodzic (Parent)	brak	element.parentNode
Przodek (Ancestor)	brak	element.parentNode.parentNode...
Poprzedni sąsiad (Previous sibling)	brak	element.previousSibling
Następny sąsiad (Next sibling)	element + sąsiad	element.nextSibling
Pierwsze dziecko (First child)	element > dziecko:first-child	element.firstChild
Ostatnie dziecko (Last child)	element > dziecko:last-child	element.lastChild
Wszystkie dzieci (Children)	element > *	element.childNodes
Potomkowie (Descendants)	element *	element.getElementsByTagName("")

5

Metody DOM

- selektory CSS nie dają możliwości wybrania przodków ani poprzedników aby ułatwić przeglądarkom wyświetlanie progresywne (element występujący później w dokumencie nie powinien mieć wpływu na wcześniejsze) oraz uniemożliwić zrobienie kombinacji selektorów dających błędne koło

6

Wcześniejsze modele dokumentu

- DOM Netscape (DOM0) - znacznie różniący się od W3C DOM, był częścią pierwotnego języka Javascript wprowadzonego przez Netscape. Dzielił dokument na kolekcje obrazków, linków, formularzy, itp. Część z tych rozwiązań została zachowana w W3C DOM dla kompatybilności wstecz
- DOM Microsoft (DHTML) document.all był zbiorem wszystkich elementów posortowanym wg ich identyfikatora, kolejności w dokumencie lub nazwy tagu. Microsoft porzucił swój DOM na rzecz tego z W3C

7

DOM dla JavaScript

- DOM - sposób przedstawienia dokumentu; zestaw metod i pól, które umożliwiają odnajdywanie, zmienianie, usuwanie i dodawanie elementów
- DOM W3C (dwie części):
 - **podstawowy** (ang. **core**) - ogólny sposób reprezentowania dokumentów XML; przedstawia dokument jako drzewo zawierające węzły (ang. node). Każdy węzeł może być elementem, fragmentem tekstu, komentarzem, instrukcją preprocesora (np. dołączonym fragmentem PHP) albo encją
 - DOM HTML (typowy dla przeglądarek) - to zestaw metod ułatwiających tworzenie dynamicznych stron oraz zapewniających kompatybilność wstecz z wcześniejszym prostym DOM Netscape. DOM HTML przedstawia dokument jako kilka kolekcji obiektów określonych typów (np. formularze, obrazki) oraz dodaje pola/metody ułatwiające dostęp do funkcjonalności specyficznej dla HTML, jak np. odczyt pól formularza.

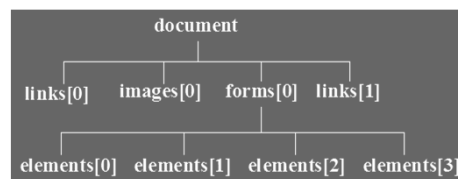
8

Podstawowe obiekty

- **window** - główny, globalny obiekt w DOM; w nim przechowywane są wszystkie globalne zmienne i funkcje a także obiekt document
- **document** - główny element dokumentu reprezentuje całą stronę HTML/XHTML
- **document.root** - korzeń dokumentu
- **document.body** - ciało dokumentu

9

Zależności hierarchiczne tablic DOM HTML

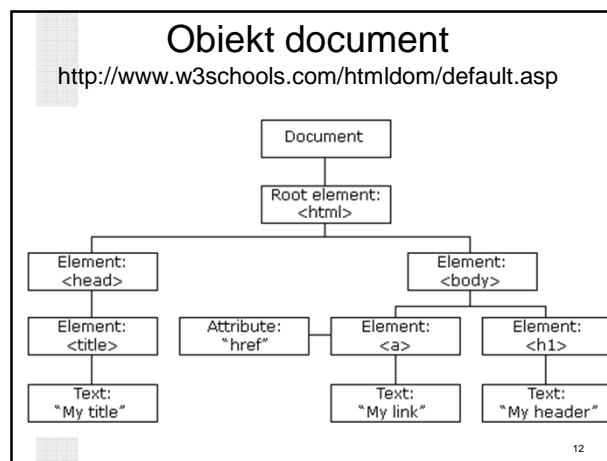


- Odwołanie do wartości pola formularza:
`document.forms[0].elements[1].value`

10

Dokument HTML

11




Odnajdywanie elementów

- `getElementById(id)` - po identyfikatorze 'id'
`var elid = document.getElementById('elid')`
`if (!elid) alert('nie ma elementu o tym id!')`
- `getElementsByName(name)` - po nazwie 'name' elementu
- `getElementsByTagName(tag)` - po nazwie taga; zwraca kolekcję wszystkich 'tagów' o określonej nazwie; * zamiast nazwy - wszystkie elementy; wywołanie metody na dowolnym elemencie - zwraca tylko elementy w nim zawarte, np.:
`var tab = elid.getElementsByTagName('nazwa')`
`for(i=0;i<tab.length;i++) tab[i]`
tab - **kolekcja** (tablica) wszystkich elementów o nazwie **nazwa** wewnątrz elementu reprezentowanego przez obiekt **elid**; **kolekcje** reagują na zmiany w dokumencie - jeśli element zostanie usunięty z dokumentu, to zniknie także ze wszystkich kolekcji

13

Przykład – getElementByName



```

<head>
<script type="text/javascript">
function getElements()
{ var x=document.getElementsByName("tekst");
  alert(x.length);
}
</script>
</head>
<body>
<input name="tekst" type="text" size="20" /><br />
<input name="tekst" type="text" size="20" /><br />
<input name="tekst" type="text" size="20" /><br />
<br />
<input type="button" onclick="getElements()"
value="Ile jest elementow o nazwie 'tekst'?" />
</body></html>

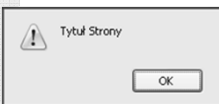
```

14

Przykład – getElementById

Tytuł Strony

Kliknij na tytuł strony i pobierz jego zawartość



```

<head>
<script type="text/javascript">
function getValue()
{ var x=document.getElementById("Baner")
  alert(x.innerHTML)
}
</script>
</head>
<body>
<h1 id="Baner" onclick="getValue()">Tytuł
Strony</h1>
<p>Kliknij na tytuł strony i pobierz jego
zawartość</p>
</body></html>

```

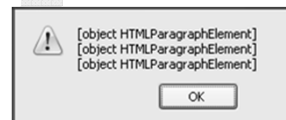
15

Przykład – getElementsByTagName

Akapit 1

Akapit 2

Akapit 3



```

<head>
<script type="text/javascript">
function getTag()
{
  var akapit =
    document.getElementsByTagName("p");
  var n=akapit.length;
  var napis="";
  for (i=0;i<n;i++) napis+=akapit[i]+"n";
  alert(napis);
}
</script></head>
<body onload="getTag()">
<p>Akapit 1</p>
<p>Akapit 2</p>
<p>Akapit 3</p>
</body></html>

```

16

Przykład – formularze obliczeniowe

Dodawanie

Liczba 1:

Liczba 2:

Oblicz sumę:

```

... <script type="text/javascript">
function oblicz()
{ var l1=document.getElementById("l1");
  l1=l1.value; l1=parseInt(l1);
  var l2=document.getElementById("l2");
  l2=parseInt(l2.value);
  var s=document.getElementById("suma");
  s.value=l1+l2;
} </script> </head>
<body> <div><h3>Dodawanie</h3>
<table><tbody>
<tr><td>Liczba 1:</td><td><input id="l1" /></td> </tr>
<tr><td>Liczba 2:</td><td><input id="l2" /></td> </tr>
<tr><td><button onclick="oblicz()">Oblicz sumę :
<td><input id="suma" disabled="disabled"
value="" /></td></tr>
</tbody></table></div></body></html>

```

17

Skakanie po węzłach (elementach)

Każdy węzeł (czyli element, fragment tekstu, komentarz) posiada pola wskazujące na jego sąsiednie węzły

Nazwa pola	Element
firstChild	Pierwszy węzeł zawarty w tym elemencie
lastChild	Ostatni węzeł zawarty w tym elemencie
previousSibling	Sąsiedni węzeł przed tym elementem
nextSibling	Sąsiedni węzeł za tym elementem
parentNode	Element, w którym zawarty jest ten element
childNodes	Kolekcja wszystkich węzłów zawartych w tym elemencie

18

Typy węzłów - nodeType

Wartość	Typ węzła
1	Element
2	Atrybut
3	Węzeł tekstowy
4	Sekcja CDATA
5	Referencja do encji
6	Węzeł encji
7	Instrukcja przetwarzania XML
8	Komentarz XML
9	Węzeł dokumentu XML
10	Definicja typu dokumentu
11	Część dokumentu XML
12	Notacja XML

19

Wyszukiwanie tylko elementów

- Domyślnie pola wskazują na dowolne węzły, łącznie z tekstem i komentarzami w dokumencie
- Elementy można odróżnić od innych węzłów za pomocą pola nodeType:
if (element.nextSibling && element.nextSibling.nodeType == 1)
alert('sąsiedni węzeł jest elementem');

20

Atrybuty

- `getAttribute(nazwa)` – odczyt (null jeżeli atrybut nie istnieje)
- `setAttribute(nazwa, wartość)` – ustawianie
- `removeAttribute(nazwa)` – usuwanie
- `hasAttribute(nazwa)` - sprawdzenie czy dany atrybut istnieje

Np.:

```
<a href="http://kurs.browsehappy.pl" rel="archive" id="kurs">
  Kurs BrowseHappy</a>
a = document.getElementById('kurs');
var aRel = a.getAttribute('rel');
alert(aRel); //zwróci wartość atrybutu rel
if ( a.hasAttribute('class') )
  var aClassName = a.getAttribute('class');
alert(aClassName); //jeżeli istnieje class, to zwróci jego wartość
a.setAttribute("rel", "kurs"); //nadpisze rel
a.setAttribute("lang", "pl"); //stworzy nowy atrybut i
                             // przypisze mu wartość
```

21

Usuwanie, tworzenie i dodawanie elementów

- `removeChild` - metoda pozwalająca (każdemu elementowi) usunąć jeden z węzłów w nim zawartych np.:
`element.parentNode.removeChild(element)`
- element usunięty z dokumentu nie jest całkowicie niszczone i może zostać ponownie dołączony do dokumentu
- `document.createElement(nazwa)` – tworzy element, który nie jest połączony z dokumentem - trzeba go dodać za pomocą `appendChild` lub `insertBefore`
- `document.createTextNode("tekst")` - tworzy węzeł tekstowy; podobnie jak elementy, żeby stał się widoczny, trzeba go wstawić do dokumentu

22

Tworzenie, dodawanie i przenoszenie elementów - przykład

- Np.

```
var p = document.createElement('p');
document.body.appendChild(p);
var h1 = document.createElement('h1');
document.body.insertBefore(p, document.body.firstChild)
```
- Komentarz:
 - element `<p>` podany jako argument metody `appendChild(p)` dołączany jest jako dziecko na koniec elementu `<body>`, z którego metoda została wywołana
 - metoda `insertBefore` przyjmuje dwa argumenty — element do wstawienia oraz element, przed którym ma wstawić nowy. Jeśli jako drugi argument poda się null, to zadziała tak samo jak `appendChild`
 - wykonanie `appendChild` lub `insertBefore` na elementach, które są już w dokumencie przeniesie je w nowe miejsce

23

Przykład 1 – JavaScript i DOM

```
<head>
<title> DOM i JavaScript </title>
<script type="text/javascript" src="js_dom.js">
</script>
</head>
```

```
<body onload="utworz_liste1()">
  <p>Lista kolorów:</p>
  <div id="lista_kol"> </div>
</body>
```

Lista kolorów:

- ◆ Czarny
- ◆ Pomarańczowy
- ◆ Różowy

24

Przykład 1 – JavaScript i DOM

Plik js_dom.js

```
function utworz_liste1()
{ // tworzenie kodu HTML
  var lista;
  lista = "<ul>" + "<li>Czarny</li>"
    + "<li>Pomarańczowy</li>" + "<li>Różowy</li>"
    + "</ul>";
  // pobranie odwołania do elementu <div> na stronie
  elDiv = document.getElementById("lista_kol");
  // dodanie zawartości do elementu <div>
  elDiv.innerHTML = lista;
}
```

25

Przykład 1 – JavaScript i DOM

Plik js1_dom.js

```
function utworz_liste2()
{ // tworzy pierwszy węzeł tekstowy
  oHello = document.createTextNode("Lista kolorów:");
  // tworzy element <ul>
  oUl = document.createElement("ul")

  // tworzy pierwszy element <li> i dodaje do niego węzeł tekst.
  oLiBlack = document.createElement("li");
  oBlack = document.createTextNode("Czarny");
  oLiBlack.appendChild(oBlack);

  // tworzy drugi element <li> i dodaje do niego węzeł tekstowy
  oLiOrange = document.createElement("li");
  oOrange = document.createTextNode("Pomarańczowy");
  oLiOrange.appendChild(oOrange);
}
```

26

Przykład 1 – JavaScript i DOM

Plik js1_dom.js

```
// tworzy trzeci element <li> i dodaje do niego węzeł tekstowy
oLiPink = document.createElement("li");
oPink = document.createTextNode("Różowy");
oLiPink.appendChild(oPink);

// dodaje elementy <li> do elementu <ul> jako jego potomstwo
oUl.appendChild(oLiBlack);
oUl.appendChild(oLiOrange);
oUl.appendChild(oLiPink);

// pobiera odwołanie do elementu <div> na stronie
elDiv = document.getElementById("lista_kol");
// dodaje zawartość do elementu <div>
elDiv.appendChild(oHello);
elDiv.appendChild(oUl);
}
```

27

Przykład 2 – DOM, JS i CSS

```
<head>
<script type="text/javascript" src="csstest.js"></script>
<link href="styles.css" type="text/css" rel="stylesheet"/>
</head>
<body>
<table id="table"><tbody>
<tr><th id="tableHead">Technologie internetowe</th> </tr>
<tr><td id="tableFirstLine"> XHTML/CSS </td> </tr>
<tr><td id="tableSecondLine"> JavaScript </td> </tr>
</tbody> </table>
<div>
<input type="button" value="Ustaw Styl 1" onclick="setStyle1();" />
<input type="button" value="Ustaw Styl 2" onclick="setStyle2();" />
</div>
</body>
```

28

Przykład 2 – csstest.js

```
// Zmiana stylu tabeli na styl 1
function setStyle1() {
  // pobranie odwołań do elementów HTML
  oTable = document.getElementById("table");
  oTableHead = document.getElementById("tableHead");
  oTableFirstLine = document.getElementById("tableFirstLine");
  oTableSecondLine = document.getElementById("tableSecondLine");
  // ustawienie stylu
  oTable.className = "Table1";
  oTableHead.className = "TableHead1";
  oTableFirstLine.className = "TableContent1";
  oTableSecondLine.className = "TableContent1";
}
```

29

Przykład 2 – csstest.js

```
// Zmiana stylu tabeli na styl 2
function setStyle2() {
  // pobranie odwołań do elementów HTML
  oTable = document.getElementById("table");
  oTableHead = document.getElementById("tableHead");
  oTableFirstLine = document.getElementById("tableFirstLine");
  oTableSecondLine = document.getElementById("tableSecondLine");
  // ustawienie stylu
  oTable.className = "Table2";
  oTableHead.className = "TableHead2";
  oTableFirstLine.className = "TableContent2";
  oTableSecondLine.className = "TableContent2";
}
```

30

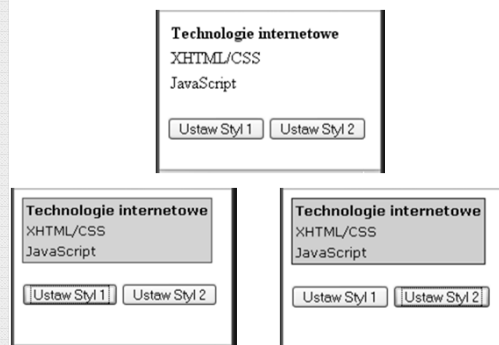
Przykład 2 – styles.css

```
.Table1 {border: DarkGreen 1px solid;
        background-color:LightGreen;}
.TableHead1 { font-family: Verdana, Arial; font-weight: bold;
               font-size: 10pt; }
.TableContent1{ font-family: Verdana, Arial; font-size: 10pt; }

.Table2 {      border: DarkBlue 1px solid;
               background-color:LightBlue;}
.TableHead2 {  font-family: Verdana, Arial; font-weight: bold;
               font-size: 10pt; }
.TableContent2{ font-family: Verdana, Arial; font-size: 10pt; }
```

31

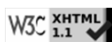
Przykład 2 – wynik działania



32

Przykład 3 - DOM, JS i CSS

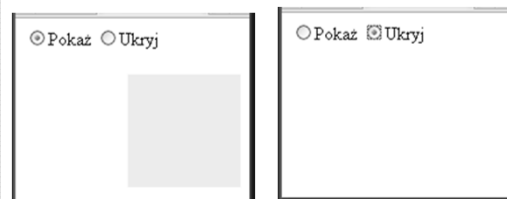
```
...
<link rel="stylesheet" href="p3.css" type="text/css" />
<script type="text/javascript">
<!-- Ukrycie JavaScriptu
function pokaz()
{ document.getElementById("wr1").style.visibility = "visible"; }
function ukryj()
{ document.getElementById("wr1").style.visibility = "hidden"; }
//Koniec kodu JavaScript -->
</script></head>
<body>
<div id="wr1" > </div>
<div>
<input type="radio" name="rd1" onclick="pokaz();" checked="checked" /> Pokaż
<input type="radio" name="rd1" onclick="ukryj();" /> Ukryj
</div>
</body></html>
```



33

Przykład 3 – p3.css

```
#wr1 { position:absolute; left:100px; top:50px;
       width:100px; height:100px;
       background-color:yellow;
       visibility:visible;
}
```



34