ZAJĘCIA SPECJALISTYCZNE

JavaScript

TEMAT 2-4: Obiekt „Array”.

Autor dokumentu: Wojciech Galiński | wtorek, 5 lutego 2013 r. | 312[01]/T,SP/MENiS/2004.06.14

ŹRÓDŁA WIEDZY: http://pl.wikipedia.org/, http://webmaster.helion.pl/index.php/kurs-javascript, http://krook.org/jsdom/, http://www.dynamicdrive.com/, http://www.w3schools.com.

Zagadnienia obowiązkowe

1. **Obiekt „Array”** – posiada następujące składowe:
   - ➔ **WŁAŚCIWOŚCI** – wyróżniamy:
     - ✔ **`length`** – przechowuje aktualny rozmiar (liczbę elementów) tablicy.

       PRZYKŁAD:
       ```
       tab = [1, 2, 3];   alert("Rozmiar tablicy = " + tab.length);
       ```
   - ➔ **METODY** – wyróżniamy:
     - ✔ **`concat`** – powoduje przyłączenie do obiektu wartości podanych w argumentach.

       PRZYKŁAD:
       ```
       var tab1 = [1, 2, 3], tab2 = [6, 7, 8],
       tab3 = tab1.concat(3, 2, 1), tab4 = tab1.concat(4, 5, tab2, 9);
       document.write(tab3 + '<br />' + tab4);
       ```
     - ✔ **`indexOf(wartosc)`**, **`lastIndexOf(wartosc)`** – (nie działa w IE) wyszukuje **pierwsze / ostatnie** wystąpienie wartości „wartosc” w tablicy.

       PRZYKŁAD:
       ```
       alert(tab3 + '-' + tab3.indexOf(2) + '-' + tab3.lastIndexOf(2));
       ```
     - ✔ **`join(`*`separator`*`)`** – łączy elementy tablicy w tekst składający się z wartości tablicy rozdzielonych separatorem (domyślny separator to **przecinek**).

       PRZYKŁAD: `document.write('[' + tab3.join('][') + ']');`
     - ✔ **`toString()`**, **`toLocaleString()`** – obie metody są zwykle równoważne metodzie `join(",")`, ale czasami wartość funkcji składowej „toLocaleString” jest zależna od języka wybranego w przeglądarce internetowej, a nawet od samej przeglądarki.
     - ✔ **`pop()`**, **`push(wartosc1`**`, …, wartoscN`**`)`** – obsługa tablicy jako stosu:
       - pierwsza usuwa ostatni element z tablicy (stosu) i zwraca go jako wartość funkcji;
       - druga kładzie wartości podane argumentami na koniec tablicy (wierzchołek stosu).

       PRZYKŁAD:
       ```
       var tab = [1, 2];   tab.push(4,8);   document.write(tab + '|');
       document.write(tab.pop() + ' ' + tab.pop() + '|' + tab);
       ```
     - ✔ **`shift()`**, **`unshift(wartosc1`**`, …, wartoscN`**`)`** – obsługa tablicy jako kolejki:
       - pierwsza usuwa pierwszy element z tablicy (kolejki) i zwraca go jako wartość funkcji;
       - druga wstawia wartości podane argumentami na początek tablicy (kolejki).
     - ✔ **`slice(indeks_start`**`, indeks_stop`**`)`** – klonuje fragment tablicy: indeksy od **indeks_start** do **indeks_stop - 1**. Jeśli drugi argument zostanie pominięty, klonujemy do końca tablicy. Jeśli **indeks_start** lub **indeks_stop** jest ujemny, odliczamy od końca tablicy.

PRZYKŁADY:

```
var tab = [1, 2, 3, 4, 5];   document.write(tab.slice(0, 2));
document.write(' ' + tab.slice(2, -1) + ' ' + tab.slice(4, 5);
document.write(' ' + tab.slice(3) + ' ' + tab.slice(-3));
```

- ✔ **splice(indeks, liczba, wartosc1, …, wartoscN)** – podmienia zawartość we fragmencie tablicy, tzn. najpierw usuwa „**liczba**” wartości z tablicy poczynając od indeksu „**indeks**”, a następnie wstawia w to miejsce wartości „**wartosc1, …, wartoscN**”.

  PRZYKŁADY:

```
var tab = [1, 2, "x", "x", 5],  tab2 = [5, 4, 3, 2];
tab.splice(2, 2, 3, 4);   document.write(tab);
tab.splice(0, 0, tab2);   document.write('|' + tab);
```

- ✔ **reverse** – odwraca kolejność elementów w tablicy: pierwsza komórka staje się ostatnia, druga – przedostatnia, itd.

  PRZYKŁAD: `var tab = [1, 2, 3, 4];  document.write(tab.reverse());`

- ✔ **sort(*nazwa_funkcji*)** – sortowanie danych w tablicy (standardowo elementy traktujemy jako tekst). Można też użyć następująco zdefiniowanej funkcji porównującej:

```
function nazwa_funkcji(element1, element2)
{
  // treść funkcji
  return wartosc; // -1, 0 albo 1
}
```

  która zwraca: **-1** gdy **element1<element2**, **0** gdy **element1==element2**, **1** gdy **element1>element2**.

  PRZYKŁADY:

```
function porownaj_liczby(wartosc1, wartosc2)
{ return wartosc1 - wartosc2; }
tab = [5,22,7,11,1,2];
tab.sort();   document.write(tab + '<hr />');
tab.sort(porownaj_liczby);   document.write(tab + '<hr />');
```

Zadania

1. Utwórz tablicę i wypełnij ją 10 losowymi liczbami jednocyfrowymi. Posortuj te liczby.
2. Na liście z zadania pierwszego znajdź pierwszą liczbę 5 i zamień ją na liczby „7, 6, 5” i posortuj całą tablicę w porządku malejącym. Następnie odwróć kolejność elementów w tablicy. Wyświetl zawartość tablicy przed i po odwróceniu kolejności jej elementów.
3. Wpisuj z klawiatury umówione nazwy do momentu, gdy podany zostanie pusty ciąg znaków. Następnie posortuj te wpisy od Z do A. Wyświetl te nazwy oddzielając je znakiem „|”.
4. Z listy z nazw z poprzedniego zadania skopiuj 3 kolejne elementy pomijając pierwszy element tablicy.
5. Korzystając z obiektu „Array” i metod „shift” i „unshift” generujemy kolejkę składającą się z 10-20 osób. Sprawdź, jak długo czeka ostatnia osoba w kolejce zakładając, że długość obsługiwania klienta to losowa liczba 15-120 sekund, a co jakiś czas jedna osoba obsługiwana jest poza kolejnością (wskakuje na początek kolejki).
6. Korzystając z obiektu „Array” i metod „pop” i „push” wykonaj następującą grę: losujemy 3 kolorowe kulki i wrzucamy po kolei do tuby. Należy zgadywać kolory wyciąganych (w odwrotnej kolejności kulek). Jeśli uda się zgadnąć kolory kulek to w następnej rundzie liczba kulek zwiększa się o 1.
7. Korzystając z obiektu „Array” i metod „pop” i „push” przedstaw rozwiązanie problemu „Wież Hanoi” opisanego m. in. w Wikipedii.