ZAJĘCIA SPECJALISTYCZNE

JavaScript

TEMAT 2-3: Obiekt „String”.

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 „String”** - służy do obsługi tekstu (ciągów znaków pomiędzy apostrofami albo cudzysłowami). Metody działają na obiekcie, który je wywołuje.

   Posiada następujące elementy składowe:

   ➔ **WŁAŚCIWOŚCI** – wyróżniamy:

   ✔ **`length`** – przechowuje aktualną liczbę znaków w tekście, np.

   ```
   var tekst = "abc", tekst2 = new String ("xyz");  tekst2 += tekst;
   document.write("ij".length + '|' + tekst2.length);
   ```

   ➔ **METODY FORMATUJĄCE CIĄG** – (uznane za prymitywne i rzadko się je stosuje) wyróżniamy:

   ✔ **`anachor(nazwa)`**, **`link(adres)`** – pierwszy tworzy etykietę, a drugi – odsyłacz do początku lub określonego miejsca w dowolnym dokumencie HTML, np.

   ```
   var tekst = '<p>', etykieta = "".anchor("et"),
       link1 = "Na dół".link("#et"),   link2 = "Do góry".link("#");
   for (var i=0; i<2000; ++i) tekst += "ble ";   tekst += '</p>';
   document.write(link1 + tekst + tekst + etykieta + link2 + tekst);
   ```

   ✔ **`sub()`**, **`sup()`** – ustawiają tekst jako indeks **górny / dolny**, np.

   ```
   document.write("2"+"5x+1".sup() + ' ' + "x"+"1".sub()+"2".sup());
   ```

   ✔ **pozostałe** (big, blink, bold, fixed, fontcolor, fontsize, italics, small, strike) – uznane za całkiem przestarzałe i nie są używane (zostały przejęte przez style CSS).

   ➔ **METODY PRZETWARZAJĄCE CIĄGI** – wyróżniamy:

   ✔ **`charAt(indeks)`**, **`charCodeAt(indeks)`** – zwraca **znak / kod znaku** umieszczonego tekście na pozycji o numerze „indeks” (liczone od 0), np.

   ```
   document.write("abc".charAt(0) + ' ' +"abc".charCodeAt(0));
   ```

   ✔ **`concat(string1, …, stringN)`** – dołącza do bieżącego tekstu ciągi znaków przekazane argumentami (w praktyce w tym celu używa się operatora „+”), np.

   ```
   var s = "Ala";  document.write(s.concat(' ', "i kot"));
   ```

   ✔ **`String.fromCharCode(kod1, …, kodN)`** – służy do tworzenia tekstu z pojedynczych kodów znaków (funkcja statyczna – wymaga użycia nazwy klasy), np.

   ```
   document.write(String.fromCharCode(97, 98, 99, 100));
   var tekst = String.fromCharCode(120, 121, 122);   alert(tekst);
   ```

   ✔ **`indexOf(tekst)`**, **`lastIndexOf(tekst)`** – szuka **pierwszego / ostatniego** ciągu „tekst”, przesuwając się w **prawo / lewo**, np.

   ```
   var s = "abcxyzabc";
   document.write( s.indexOf("bc") + '<br />' + s.lastIndexOf("bc")
                   + '<br />' + s.indexOf("bc", 2)
                   + '<br />' + s.lastIndexOf("bc", 2) );
   ```

- ✔ s.**localeCompare(tekst)** – ciąg „**tekst**” jest porównywany (zgodnie z porządkiem alfabetycznym) z ciągiem „s”, który wywołał tę metodę. Uwzględnia się ustawienia narodowe i językowe przeglądarki.

  Metoda zwraca: **0**, gdy „**s**”==”**tekst**”; wartość **<0**, gdy „**s**”<„**tekst**”; wartość **>0**, gdy „**s**”>„**tekst**”, np.

```
alert( "ab".localeCompare("cd") + ' ' + "dc".localeCompare("ba") + ' '
       + "abc".localeCompare("abc") + ' ' + "10".localeCompare(010) );
```

- ✔ **slice(poczatek**, *koniec***)**, **substring(indeks_1**, *indeks_2***)** – zwraca fragment tekstu, który wywołał tę metodę. Parametry podają indeksy: początku i końca tego fragmentu (liczby ujemne oznaczają, że liczymy od końca tekstu), np.

```
var s = "domeczek";   document.write(s.slice(4) + ' ');
document.write(s.slice(0, -5) + ' ' + s.slice(2, 6));
```

- ✔ **split(separator**, *limit***)** – dzieli tekst na fragmenty i wstawia je jako elementy tablicy używając podanego separatora. Można ograniczyć liczbę fragmentów (licząc od początku) poprzez parametr „limit”.

  PRZYKŁADY:
```
document.write("2012-5-31".split('-') + ' ');
document.write("abcdef".split('', 4));
```

- ✔ **substr(indeks**, *liczba_znakow***)** – zwraca fragment tekstu o długości „liczba_znakow” i początku w indeksie „indeks” lub tekst od indeksu „indeks” do końca tekstu (gdy pominiemy drugi parametr).

  PRZYKŁAD: `document.write("malec".substr(1, 3));`

- ✔ **toLowerCase**, **toLocaleLowerCase**, **toUpperCase**, **toLocaleUpperCase** – zamienia litery: duże na małe oraz małe na duże (uwzględniając polskie znaki).

  PRZYKŁAD: `document.write("ZAŻÓŁĆ gęślą JAŹŃ".toUpperCase());`

➔ **METODY PRZETWARZAJĄCE CIĄGI WYKORZYSTUJĄCE WYRAŻENIA REGULARNE** – wyróżniamy:

- ✔ **search(tekst albo wyrazenie_regularne)** – zwraca indeks początku pierwszego dopasowania.
- ✔ **match(tekst albo wyrazenie_regularne)** – zwraca wszystkie dopasowania do wzorca;
- ✔ **replace(tekst albo wyrazenie_regularne, tekst_do_zamiany)** – zwraca tekst z wszystkimi dopasowaniami zamienionymi na tekst „tekst_do_zamiany”;

  PRZYKŁADY:
```
var tekst = "Abrakadabra", regexp = /abra/gi;
document.write(tekst.match(regexp) + '<br />');
document.write(tekst.replace(regexp, "abla") + '<br />');
document.write(tekst.search(regexp) + '<br />');
```

Zadania

1. Wczytaj liczbę z klawiatury i wyświetl ją w postaci binarnej ze stałą liczbą cyfr.
2. Wylosuj liczbę z przedziału 0-1E12 i wyświetl ją w postaci binarnej – zignoruj początkowe zera.
3. Zamień następujące kody liczbowe na tekst: 79, 108, 97.
4. Zamień tekst podany z klawiatury na kody liczbowe.
5. Wyświetl od tyłu tekst podany z klawiatury.
6. Znajdź indeks początku kodu pocztowego w tekście: „ABC, 48-300 NYSA, ul. Polna 2/1”.
7. W tekście „Abrakadabra” zamień wszystkie wystąpienia litery „a” (duże i małe litery) na „?”.
8. Sprawdź, które słowo będzie stało wcześniej według porządku alfabetycznego: ślimak, czy ul.