Інтернет програмування

лек 5

Методи об’єкту String

Зав.каф.КН, PhD Граф М.С.

План лек

1. Метод getElementById()
2. Властивість innerHTML
3. Об'єкт String
4. charCodeAt
5. String.fromCharCode
6. charAt
7. Concat
8. Search
9. Match
10. toLowerCase, toUpperCase
11. Substring
12. Slice
13. indexOf
14. Substr
15. replace

Метод getElementById()

```
<input type='text' id='example01' name='test' />
```

Формально getElementById() не самостійна функція, а метод об'єкта документа.

```
document.getElementById('example01')
document.getElementById('example01').value
```

Доступ до елементів сторінки можливий лише за умови унікального значення атрибута id.

Властивість innerHTML

Використовується для визначення вмісту сторінки, надаючи доступ до вмісту елементів. Воно дозволяє здійснювати як читання, а й запис інформації. Достатньо призначити цій властивості рядок тексту, і вміст контейнера елемента буде замінено на новий.

- застосовується лише для завдання значень таких елементів, як div, span, p та інших контейнерів$
- властивість innerHTML повністю переписує попереднє значення атрибута;
- не можна додати значення до innerHTML.

Об'єкт String

```
var a = 'my string'
```

```
var str = "string literal"
```

Можна вказати юнікодний символ явно через його код.

```
var str = "\u1234"
```

Властивості

Для null-рядка довжина дорівнюватиме 0.

Виведення цифри 8 у діалоговому вікні Alert:

```
var x="Netscape"
alert("The string length is " + x.length)
"12345".length // 5
```

Об'єкт String

Методи

```
var arr = str.split([separator][, limit]);
```

```
arr = "a,b,c".split(',')  // масив ["a", "b", "c"]
```

Якщо separator не вказаний або збігів немає, то масив, що повертається, складається з одного елемента - вихідного рядка.

Якщо separator є регулярним виразом з вкладеними дужками, то щоразу при збігу separator всі результати захоплення дужками, включаючи undefined результати, додаються у вихідний масив.

Якщо рядок порожній, то split повертає масив з одного порожнього рядка, а не порожній масив.

Об'єкт String

Приклад: за регулярним виразом

```
var tags = "Літо, зима, осінь 2022 року"
var re = /\s*,\s*/
var tagList = tags.split(re);
alert(tagList) //  ["Літо”, “зима”, “осінь 2022 року”"]
```

Приклад: Обмеження масиву

```
var str ="a b c d"
str.split(' ',2)
// ["a", "b"], а не ["a" "b c d"]
```

Приклад: Вкладені дужки

```
var myString = "Hello 1 word. Sentence number 2."
var splits = myString.split(/(\d)/)
// splits =  ["Hello ", "1", " word. Sentence number ", "2", "."]
```

charCodeAt

var codepoint = string.charCodeAt(index)

Юнікодне значення від 0 до 1,114,111. Перші 128 значень Unicode збігаються з кодуванням ASCII.

Приклад: Використання charCodeAt

```
// поверне 65, unicode-код для A
"ABC".charCodeAt(0) // 65
```

String.fromCharCode

String.fromCharCode(num1, ..., numN)

num1, ..., numN – послідовність чисел, що є кодами Unicode-символів

Приклад: Використання fromCharCode

```
// поверне рядок "ABC".
String.fromCharCode(65,66,67)
```

charAt

str.charAt(index)

Приклад: перерахуванн символів у рядку

```
var anyString="Привіт, мир"
alert("Символ на позиції 0 '" + anyString.charAt(0) + "'");
alert("Символ на позиції 3 '" + anyString.charAt(3) + "'");
alert("Символ на позиції 77 '" + anyString.charAt(999) + "");
```

Виведе:

```
Символ на позиції 0 'П'
Символ на позиції 3 'в'
Символ на позиції 77 ''
```

concat

result = string1.concat(string2, string3[, ..., stringN])

Аргументи

string2...stringN – терміни, які будуть додані до string1

Виконує дію, аналогічну оператору + для рядків.

```
a = "рядок"
b = " дуже"
c = " довгий"
alert(a.concat(b,c)) // рядок дуже довгий
```

lastIndexOf

str.lastIndexOf(searchValue[, fromIndex])

Аргументи

searchValue – шуканий підрядок
fromIndex – індекс, від якого починати пошук до початку рядка. Від 0 до str.length-1. Значення за замовчуванням: str.length

Символи у рядку йдуть зліва направо. Позиція першого символу 0, останнього – довжина рядка мінус один.

```
"canal".lastIndexOf("a")    // 3
"canal".lastIndexOf("a",2) // 1
"canal".lastIndexOf("a",0) // -1
"canal".lastIndexOf("x")    // -1
```

Приклад: регістр важливий

```
"Привіт".lastIndexOf("п") // -1
```

search

idx = str.search(regexp)

Аргументи

regexp – об'єкт типу RegExp або рядок

Приклад: Перевірка на збіг

```
function testinput(re, str){
    if (str.search(re) != -1)
        return true
    else
        return false
}
alert(testinput(/something/, "somestr")) // false
```

match

str.match(reg) **str.match(regexp)**

Аргументи **regexp** - регулярний вираз у вигляді об'єкта RegExp або рядок

Якщо регулярне вираження без прапора "g", то повертає той самий результат, як regexp.exec(str).
Якщо ж для регулярного виразу вказано прапор "g", то метод повертає масив, що містить усі збіги.
Якщо збігів немає, метод повертає не порожній масив, а null.

Пример без глобального поиска

```
str = "За інформацією звернулись: Розділ 3.4.5.1";
re = /розділ (\d+(\.\d)*)/i
found = str.match(re)
document.write(found)
```

Приклад із глобальним пошуком

```
str = "За інформацією звернулись: Розділ 3.4.5.1, Розділ 7.5";
re = /глава (\d+(\.\d)*)/ig
found = str.match(re)
alert(found)
```

toLowerCase, toUpperCase

Синтаксис
str.toLowerCase()

```
var upperText="ПриВіт"
document.write(upperText.toLowerCase()) // привіт
```

Синтаксис
str.toUpperCase()

```
var upperText="ПриВіт"
document.write(upperText.toUpperCase()) // ПРИВІТ
```

substring

Аргументи

indexA – ціле число від 0 до довжини рядка-1
indexB - ціле число від 0 до довжини рядка-1

Якщо indexA = indexB, повертається порожній рядок

Якщо indexB не вказано, substring повертає символи до кінця рядка

Якщо якийсь із аргументів менший за 0 або є NaN - він вважається рівним 0

Якщо один з аргументів більше, ніж довжина рядка - він вважається рівним довжині рядка

Наприклад, str.substring(1, 0) == str.substring(0, 1).

```
var str = "Мій рядок";
alert(str.substring(0,3)); // Мій
alert(str.substring(1,2)); // і
alert(str.substring(3,0)); // Мій
alert(str.substring(4)); // "рядок"
```

slice

Синтаксис
var sub = string.slice(beginSlice[, endSlice])

Метод slice повертає частину рядка від beginSlice до endSlice, але не включаючи символ під номером endSlice, не змінюючи рядок, що викликає.

```
var str ="От такий рядок"
str.slice(0,2) // "От"
str.slice(1,-1) // "т такий рядо"
```

indexOf

Синтаксис
str.indexOf(searchValue[, fromIndex])

```
"Привіт, світ".indexOf("Привіт")    // поверне 0
"Привіт, світ".indexOf("Корова")    // поверне -1
"Привіт, світ".indexOf("світ")    // поверне 8
"Привіт, світ".indexOf("Світ") // поверне -1
```

Приклад: з позицією

```
"Привіт, світ".indexOf("Привіт", 0)    // поверне 0
"Привіт, світ".indexOf("р", 1)    // поверне 1
"Привіт, світ".indexOf("р", 5)    // поверне -1
```

Приклад: для порожнього підрядку

```
"Привіт, світ".indexOf("", 5)    // поверне 5
"Привіт, світ".indexOf("", 7)    // поверне 7
"Привіт, світ".indexOf("", 999)  // поверне 11 (довжину)
```

indexOf

приклад демонструє використання indexOf для підрахунку кількості повторень літери у рядку

```
count = 0;
pos = str.indexOf("x");
while ( pos != -1 ) {
    count++;
    pos = str.indexOf("x",pos+1);
}
```

substr

Синтаксис
var sub = string.substr(start[, length])

Метод substr починає збирати рядок із позиції start і закінчує, коли збере length або дійде до кінця рядка.
Якщо start більше або дорівнює довжині рядка, substr повертає порожній рядок.
Якщо start < 0, то позиція визначається з кінця рядка. При цьому якщо отримана позиція - до початку рядка, то як start береться 0. Ця поведінка не підтримується Internet Explorer.
Якщо параметр length не вказано - підрядок береться до кінця рядка.
Якщо length <= 0 – повертається порожній рядок.

```
var str = "abcdefghij";
alert("(1,2): "     + str.substr(1,2));   // (1,2): bc
alert("(-3,2): "    + str.substr(-3,2));  // (-3,2): hi
alert("(-3): "      + str.substr(-3));    // (-3): hij
alert("(1): "       + str.substr(1));     // (1): bcdefghij
alert("(-20, 2): " + str.substr(-20,2)); // (-20, 2): ab
alert("(20, 2): "   + str.substr(20,2));  // (20, 2):
```

replace

Синтаксис
str.replace(regexp, newSubStr|function)

Якщо ви вказуєте рядок як другий параметр - він може включати наступні спеціальні поєднання:
$$ – Вставляє "$"
$& – Вставляє підрядок, що збігся.
$` – Вставляє частину рядка, який передує підстроці, що збігся.
$' – Вставляє частину рядка, який слідує за підрядком, що збігся.
$n або $nn, де n/nn - десяткові цифри – Вставляє n-у дужку у збігу, якщо перший аргумент - об'єкт RegExp

Функція заміни

Наступний приклад поверне my XX, zz.

```
function replacer(str, p1, p2, offset, s) {
    return p1 + ", " + p2;
}
newString = "my XXzz".replace(/(X+)(z+)/, replacer)
```

Значення параметрів під час виклику replacer:

str - "XXzz" - підрядок, що збігся
p1 – "XX" - перша дужка
p2 - "zz" - друга дужка
offset – 3 - позиція в тексті для пошуку
s – "my XXzz" - рядок, що викликає, тобто рядок для пошуку

replace

Приклад: Глобальна заміна рядка

```
str = "тест ще тест"
str.replace(/тест/g,"пройшов") // = "пройшов ще пройшеов"
```

// або так

```
str.replace(new RegExp("тест",'g'),"пройшов")
```

Приклад: Заміна з посиланнями

```
var re = /(\w+)\s(\w+)/;
var str = "John Smith";
var newstr = str.replace(re, "$2, $1") // "Smith, John"
```

Методи об'єкта string: **ancor, big, blink, bold, fixed, fontcolor, fontsize, italic, link, small, strike, sub, sup**

Властивості вікна браузера clientHeight, clientWidth

Дякую за увагу!