Практична робота №7

Тема: ***Функції. Рядкові операції. DOM(Document Object Model)***

Завдання0. *Опрацювати теоретичні відомості:*

https://itchief.ru/lessons/javascript/javascript-dom-adding-and-removing-nodes
https://learn.javascript.ru/dom-nodes
https://learn.javascript.ru/dom-navigation
https://learn.javascript.ru/searching-elements-dom
https://learn.javascript.ru/basic-dom-node-properties
https://learn.javascript.ru/dom-attributes-and-properties
https://learn.javascript.ru/modifying-document
https://learn.javascript.ru/styles-and-classes

Завдання1. (*Рядкові операції)*

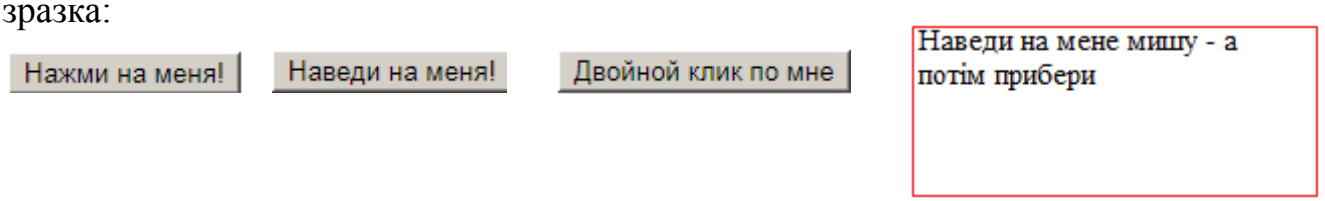
1) Знайдіть у себе на комп'ютері або в Інтернеті текст на 2000-4000 символів.
2) Вставте даний текст на сторінку.
3) Створіть текстове поле, в яке користувач буде вводити рядок, який буде потрібно знайти в тексті.
4) Напишіть скрипт, який в тексті на сторінці виділятиме всі збіги тегом <b>.

Примітка: Після введення шуканого рядка в тексті на сторінці повинні бути виділені тегом <b> всі знайдені збіги з рядком, щоб користувач відразу бачив, де знайдений його рядок. Для пошуку використовуйте методи indexOf () та slice(), також будуть потрібні цикли. Якщо збігів немає, то через alert (): «Нічого не знайдено!».

Завдання2*.(Document Object Model)*

Події через ***атрибути****:*

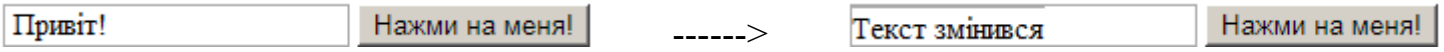
1) Створіть скрипт, по виконанні якого з'являлося б текстове повідомлення за даними зразка:



Метод ***getElementById*** *та робота з атрибутами:*

2) По натисканні на кнопку через **alert** повинно з'явитися повідомлення, яке користувач введе в поле: 

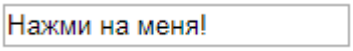
3) По натисканні на кнопку текст, який користувач введе в поле, зміниться на інший:



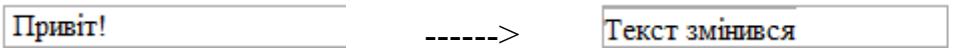
4) По натисканні на кнопку зображення зміниться:

Робота з this

5) По кліку на текстовому полі з’явиться повідомлення з цим текстом



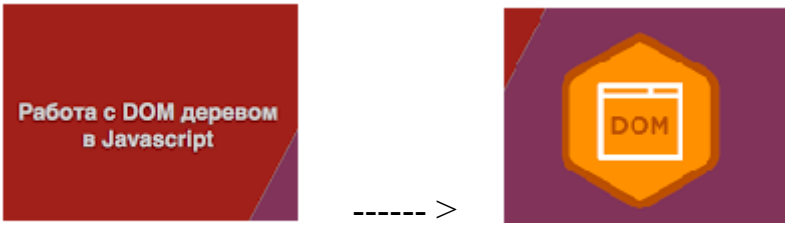
6) По кліку на текстовому полі цей текст зміниться на інший:



7) По натисканню на кнопку текст на кнопці змінюється:

8) По натисканню на кнопку кнопка стає недосяжною (властивість disabled = true/false):

9) По наведенню мишкою на зображення воно змінюється:



Завдання3. *DOM*

1) Зробіть просте HTML-меню (декілька посилань)
2) Створіть посилання «Додати» і «Видалити» за межами меню.
3) При натисканні на посилання «Додати» в меню повинен бути доданий ще 1 пункт, причому нумерація повинна зберігатися, тобто якщо було «Посилання 5», то повинно з'явитися «Посилання 6».
4) При натисканні на «Видалити» повинно видалятися останній пункт в меню.
5) Якщо видаляти вже годі й користувач натиснув «Видалити», то вивести через alert (): «Уже все видалено!».

Методичні рекомендації

Рядкові операції

Об'єкт **String** використовується для роботи з рядками.
Як і годиться, вивчення класу String починаємо з його конструктора, який і створює об'єкт String.

```
var str = new String ( "javascript");
document.write (str);
```

На початку створюється новий об'єкт викликом конструктора класу String з параметром у вигляді рядка, який ми хочемо отримати. Наступним оператором ми виводимо цей об'єкт в вікно браузера.
Визначити довжину рядка - властивість **length**:

```
document.write (str.length);
```

Тепер перейдемо до основних методів JavaScript.

Метод **charAt ()** дозволяє отримати символ по номеру в рядку. Нумерація починається з нуля, тому перший символ має індекс 0:

```
var str = new String ( "string в javascript javascript");
document.write (str.charAt (0) + str.charAt (3));
```

На початку ми створюємо екземпляр об'єкта String. Далі ми, використовуючи метод charAt () отримуємо 1-ий символ ("s"), потім з'єднуємо з 4-м ("i") і друкуємо. Вийшов рядок ("si").

Метод **indexOf ()** займається пошуком підрядка в заданій стрічці і повертає перший індекс входження. наприклад:

```
document.write (str.indexOf ( "ipt"));
```

В даному випадку Ви побачите, що повернуто буде число "16". Зверніть увагу, що якщо збігів буде знайдено декілька, то повернеться найперше. А якщо не знайдено жодного, то повернеться "-1". Також у методі **indexOf** () існує і другий необов'язковий параметр, що означає, від якого символу вести пошук:

```
document.write (str.indexOf ( "ipt", 17));
```

В даному випадку, результатом виконання скрипта буде число "27". Очевидно, що даний метод використовується в першу чергу для пошуку.

Метод **replace ()**, який застосовується при заміні підрядка. Він приймає два параметри: підрядок, який треба замінити, і підрядок, на який треба замінити (замінюється лише 1-е входження):

```
document.write (str.replace ( "javascript", "html"));
```

На виході вийде такий рядок: "string в html javascript".

Метод **slice()** дозволяє отримати з початкового рядка його частину. Він приймає два параметри: перший індекс, з якого повинен починатися отриманий рядок, і другий індекс, що означає номер останнього символу, який увійде в повернутий рядок.
Приклад:

```
document.write (str.slice (2, 5));
```

В результаті у вікні браузера з'явиться рядок: "rin". Зверніть увагу, що нумерація знову починається з нуля (і взагалі, звикайте, що нумерація завжди починається з нуля). Також зауважте, що індекс з номером "2" в результуючий рядок увійшов, а індекс з номером "5" вже не увійшов, тому що другий параметр дорівнює якраз 5.

Також у методу **slice ()** є ще один різновид. Якщо Ви не вкажете другий параметр, то буде повернутий рядок, який починається з першого індексу і до кінця заданого рядка.

```
document.write (str.slice (2));
```

В результаті вийде такий рядок: "ring в javascript javascript".

І останні два часто використовуваних методи - це **toLowerCase ()** і **toUpperCase ()**, які призводять вихідний рядок до нижнього і верхнього регістру відповідно. *Приклад*:

```
document.write (str.toLowerCase ());
document.write (str.toUpperCase ());
```

Розглянемо приклад:

```
<!doctype html>
<html>
<head>
</head>
<body>
<script type="text/javascript">
var str = "Невеликий рядок";
var text = "Довжина рядка = " + str.length;
text +="\nРядок в нижньому регістрі: " + str.toLowerCase();
text +="\nРядок в верхньому регістрі: " + str.toUpperCase();
text +="\nПідрядок від 1-го символа включно до 8-го не включно : " + str.substring(0, 7);
text +="\n3-й символ - це: " + str.charAt(2);
text +="\nПідрядок 'ряд' починається з індекса: " + str.indexOf("ряд");
text +="\nПідрядок 'hello' починається з індекса: " + str.indexOf("hello");
alert(text);
</script>
</body>
</html>
```

Результат:

```
Довжина рядка = 15
Рядок в нижньому регістрі: невеликий рядок
Рядок в верхньому регістрі: НЕВЕЛИКИЙ РЯДОК
Підрядок від 1-го символа включно до 8-го не включно : Невелик
3-й символ - це: в
Підрядок 'ряд' починається з індекса: 10
Підрядок 'hello' починається з індекса: -1
```

DOM *(Document Object Model)*

DOM –це деякий стандарт (набір властивостей і методів), що дозволяє обходити деякі елементи розмітки (html, xml, ... -розмітки). DOM є альтернативою JQUERY. До речі, JQUERY використовує DOM.

Приклад створення деякого меню, в якому буде рухатися червона підсвітка у посилань в меню:

```
<!doctype html>
```

```
<html>
<head>
	<script type="text/javascript">
	 var active = 0;
	 var links = false;

	 function parseDocument(){
	   if (!links) {
	      var nodelist = document.getElementById("menu").childNodes;
	      links = new Array();
	      var k=0;
	      for (var i = 0; i < nodelist.length; i++){
	          if(nodelist[i] instanceof  HTMLLIElement) {
	             var li  =  nodelist[i].childNodes;
	             for(var j = 0; j < li.length; j++) {
	                 if  (li[j] instanceof HTMLAnchorElement){
	                     links[k] = li[j];
	                     k++;
	                        }
	             }
	          }
	       }
	    }
	        for (var i = 0; i < links.length; i++) {
	            if (i == active) links[i].style.color = "red";
	            else links[i].style.color = "blue";
	              }
	        if ((active + 1) == links.length) active = 0;
	        else active++;
	        setTimeout(parseDocument, 100);
	   }
	</script>
</head>
<body onload = "parseDocument()">
	<ul id="menu">
	     <li>
	   <a href="#">Посилання 1</a>
	     </li>
	     <li>
	   <a href="#">Посилання 2</a>
	     </li>
	     <li>
	   <a href="#">Посилання 3</a>
	     </li>
	     <li>
	   <a href="#">Посилання 4</a>
	     </li>
	     <li>
	   <a href="#">Посилання 5</a>
	     </li>
```

```
    </ul>

</body>
</html>
```

В цьому прикладі ми оголосили дві глобальні змінні **active** та **links:**

- var **active** позначає поточний активний елемент;
- var **links** – масив, який зберігатиме всі 5 об'єктів (Посилання 1)

Завдання DOM – отримати доступ до цих об'єктів

```
var nodelist = document.getElementById("menu").childNodes;
```

- var **nodelist** – цей об'єкт має відношення до стандарту DOM, по суті це деякий масив, який містить набір елементів
- методом **getElementById**("menu") ми отримали доступ до списку <ul>...

childNodes – це дочірні вузли, тобто ми отримали доступ до кожного

```
if (nodelist[i] instanceof HTMLLIElement) {
    var li = nodelist[i].childNodes;
}
```

Тут ми перевіряємо, чи елемент масиву **nodelist** є html-елементом (все, що знаходиться між <ul>..., тобто і пробіли теж, є елементами масиву **nodelist**).

instanceof HTMLLIElement – чи належить елемент до html -**li**-елементу

var li = nodelist[i]. childNodes; - присвоюємо **li** дочірні вузли (без пробілів).

Далі ми заповнюємо масив тільки посиланнями:

```
for (var j = 0; j < li.length; j++) {
    if (li[j] instanceof HTMLAnchorElement) {
        links[k] = li[j];
        k++;
    }
}
```

тут instanceof HTMLAnchorElement – означає чи є елемент посиланням

<body onload = "parseDocument()"> - при завантаженні викликаємо функцію parseDocument()

Тепер в цій функції створюємо рухаючу червону підсвітку:

if (i == active) links[i].style.color = "red"; - якщо це поточний елемент, то він буде червоним

Далі йдемо до наступного посилання:

```
if ((active + 1) == links.length) active = 0;
                else active++;
                setTimeout(parseDocument, 100);   //для циклічного переміщення
```

```
                                    //підсвітки використовуємо таймер через кожні 100 мс
        }
```

Результат:

- Посилання 1
- Посилання 2
- Посилання 3
- Посилання 4
- Посилання 5

.....

- Посилання 1
- Посилання 2
- Посилання 3
- Посилання 4
- Посилання 5

- Посилання 1
- Посилання 2
- Посилання 3
- Посилання 4
- Посилання 5