Л7. Об’єкти, функції і події в JavaScript

Навчальні питання:

1. **Оператори маніпулювання об'єктом**
2. **Об'єктна модель JavaScript**
3. **Використання вбудованих об'єктів і функцій**

Література.

1.Флэнаган Д. JavaScript. Подробное руководство.– Пер. с англ.– СПб: СимволПлюс,

2008.– 992 с., ил.

2. Вадим Дунаев Самоучитель JavaScript

Оператори маніпулювання об'єктом

- JavaScript має способи управління об'єктами: оператори for...in, new, this і with.
- Оператор for...in привласнює змінною по черзі всі властивості об'єкту. Для кожної властивості JavaScript виконує вказані твердження. Оператор for...in виглядає таким чином:
- for (variable in object)
- { statements
- }

```
function dump_props(obj, obj_name){
    var result = ""
    for (var i in obj) {
        result+=obj_name+"."+i+"="+obj[i]+""
    }
    result += ""
    return result}
```

Оператор new

- Оператор new дозволяє створювати зразок визначуваного користувачем типу об'єкту. Використовуйте, new таким чином:
- *ObjectName = new objectType* (*param1,*
- [param2],[paramN])

Ключове слово this

- Використовуйте this, щоб звернутися до поточного об'єкту. Взагалі, this звертається до зухвалого об'єкту в методі. Використовуйте this таким чином:
-
- this [.propertyName]

Оператор with

- Оператора with встановлює об'єкт для набору тверджень. Усередині набору тверджень, будь-які посилання на властивості без явної вказівки об'єкту предпологають об'єкт за умовчанням. Оператор with виглядає таким чином:
- with (object)
- *{*
- *statements*
- *}*
-

Об'єктна модель JavaScript

- **Об'єкт** - це деяка сукупність, у яку можуть входити як примітивні, так і складені дані, включаючи функції та інші об'єкти.
- Члени-дані називаються **властивостями** об'єкта, а
- Члени-функції - **методи.**

Або: властивості - характеристики об'єкта, а методи - дії, які об'єкт може виконувати.

- Доступ до властивості об'єкта можна одержати, указавши крапку й наступне за нею ім'я властивості відразу ж після імені об'єкта. Наприклад

```
alert("Версія вашого браузера: " +
navigator.appversion)
```

- Доступ до методів об'єктів - крапка та наступне за нею ім'я але наприкінці імені методу слід додати пару круглих дужок.

```
window.close();
```

Якщо метод має аргументи, вони вказуються всередині дужок.

```
document.write("Цей текст записується в документ.");
```

Усі об'єкти, доступні сценарію мовою Javascript, підрозділяються на три групи:

- *вбудовані об'єкти* виконуючої системи;
- *об'єкти середовища*, у якому виконується сценарій (тобто або об'єкти клієнта, або об'єкти сервера);
- *користувацькі об'єкти*, створювані в процесі виконання сценарію

Об'єкти і Властивості

- Об'єкт JavaScript має властивості асоційовані з ним. Звертатися до властивостей об'єкту можна наступною простою системою позначень:
-
- objectName.propertyName

- І ім'я об'єкту, і ім'я властивості чутливі до регістра. Наприклад, хай існує об'єкт, з ім'ям myCar (ми обговоримо, як створювати об'єкти пізніше - тепер, тільки приймаємо, що об'єкт вже існує). Можна дати властивості, іменовані make, model, і year таким чином:
-
- myCar.make = "Ford"
- myCar.model = "Mustang"
- myCar.year = 69;

- Можна також звернутися до цих властивостей, використовуючи систему позначень таблиці таким чином:
-
- mycar["make"]= "Ford
- myCar["model"]= "Mustang"
- myCar["year"]= 69;

Функції і Методи

Функції - один з фундаментальних вбудованих блоків в JavaScript. Функція - JavaScript процедура - набір тверджень, які виконують певне завдання.
Визначення функції складається з ключового слова function, що супроводжується
Ім'ям функції
Списком аргументів функції, прикладеної в круглих дужках, і відокремлювані комами
JavaScript твердженнями, які визначають функцію, прикладені у фігурних дужках {...}

```
function pretty_print(string)
{ document.write(" " + string)
 }
```

Аргументи функції не обмежені тільки рядками і числами.

Аргументи функції зберігаються в таблиці. Усередині функції, можна адресувати параметри таким чином:

functionName.arguments [i]

Де functionName - ім'я функції, і i - порядкове число аргументу, що починається з нуля. Так, перший аргумент у функції, з ім'ям myfunc, буде myfunc.arguments [0]. Загальне число аргументів позначене змінним arguments.length.

Функція може навіть бути рекурсивною, тобто вона може викликати себе. Наприклад, існує функція, яка обчислює факторіали:

```
function factorial(n){
  if ((n == 0) || (n == 1))
    return 1
  else {
    result = (n * factorial(n-1))
    return result
  }
}
```

Визначення методів

- *Метод - функція, пов'язана з об'єктом. Визначається метод таким же чином, як визначається стандартна функція. Потім, використовуйте наступний синтаксис, щоб пов'язати функцію з існуючим бъектом:*
-
- *object.methodname = function_name*

```
function validate(obj, lowval, hival)
{
   if ((obj.value < lowval) || (obj.value > hival))
      alert("Invalid Value!")
}

< INPUT TYPE = "text"
    NAME = "age"
    SIZE = 3
    onChange="validate(this, 18, 99) ">
```

Користувацькі об'єкти

Існує два способи створення нових об'єктів в JS, а саме:

Використання ініціалізатора об'єкту.

Використання конструктора об'єкту.

Створення об'єктів за допомогою ініціалізатора

Цей спосіб дозволяє одночасно створити об'єкт і надати значення всім або частині його властивостей. Застосовується у випадках, коли створюється об'єкт з унікальним набором властивостей.

***{властивість:значення* [*,властивість:значення*]?}**

Наприклад, об'єкт mybrowser може бути створений так:

var mybrowser = {name: "Microsoft Internet Explorer", version: "5.5"};

Створення об'єктів за допомогою конструктора

Цей спосіб застосовується в тих випадках, коли ми прагнемо створити клас об'єктів з певним набором властивостей, а потім створювати нові об'єкти, просто вказуючи, до якого класу вони повинні належати. Для цього потрібно спочатку створити *конструктор об'єктів*

- Наприклад:

```
function Browser(name, version) {  this.name = name; this.version = version;}
```

Тепер для створення нових об'єктів класу **Browser** досить викликати цей конструктор в операції new:

```
var mybrowser = new Browser("Microsoft Internet Explorer", "5.5");
```

Вбудовані об'єкти

- Javascript містить глобальний обєкт, який є середовищем його виконуючої системи, а також наступні *вбудовані об'єкти*:

Об'єкт	Опис	Об'єкт	Опис
Array	Масиви	Math	Математичні функції й константи
Boolean	Логічні об'єкти	Number	Числові об'єкти
Date	Дата й час	Object	Прототип інших об'єктів
Error	Виключення	RegExp	Регулярні вирази
Function	Функції	String	Строкові об'єкти

Глобальний об'єкт (**Global**) створюється виконуючою системою Javascript перед початком виконання сценарію. Це єдиний об'єкт, який не має імені, і тому доступ до його властивостей і методів здійснюється без імені об'єкта(***іноді називають властивостями й методами верхнього рівня***).

Властивості глобального об'єкта	
Властивість	Опис
Infinity	Спеціальне значення "нескінченність".
NaN	Спеціальне значення "не число".
undefined	Невизначене значення.

3.Події

Подія - ця яка-небудь дія, здійснювана користувачем або браузером.

Розрізняють події генеруємі користувачем (клацання або рух миші, зміна даних форми), та браузером (закінчення завантаження документа, закриття сторінки, помилка й т.п.).

Кожна подія має ім'я : click, mouseout, focus, load

Відповідний їй обробник:onclick, onmouseout, onfocus, onsubmit, onreset, onload...

Найбільш популярні й корисні події

Події форм	**та елементів сторінки**
onchange()	Елемент втрачає фокус уведення, а вміст елемента змінився за час, поки елемент був у фокусі.
onselect()	Якась частина тексту усередині елемента стає виділеною.
ousubmit()	У формі натиснута кнопка "Відправити", але відправлення форми на сервер ще не проводилося.
Події миші	.
onclick()	Зроблено клік кнопкою миші на елементі керування. Подія виникає після того, як кнопка миші була відпущена.
onmousedown()	Натиснута кнопка миші.
onmousemove()	Покажчик миші рухається усередині області відображення елемента.
onmouseout()	Покажчик миші вийшов з області відображення елемента.
oumouseover()	Покажчик миші перебуває усередині області відображення елемента.
onmouseup()	Віджата кнопка миші.

Події вікна (об'єкт window).

1.onblur() Елемент керування втрачає фокус уведення, тобто курсор переходить до іншого елемента.

2.onfocus() Відображуваний елемент одержав фокуса введення. Для текстових полів ця подія означає, що курсор уже перебуває в даному елементі.

3.onload() Завершене завантаження сторінки.

4.onunload() Проводиться вихід з документа (закриття або перенапрямок сторінки на іншу адресу).

Події клавіатури.

1.onkeydown() Натиснута кнопки на клавіатурі.

2.onkeypress() Кнопка на клавіатурі натиснута й не відпускається довше, інтервалу повторення.

3.onkeyup() Відпущена раніше натиснута кнопка.

```
<html>
<head>
<title>Пример: Закриваємо Вікно</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<link rel="stylesheet" href="../../../../lessons.css" type="text/css">
</head>
<body bgcolor="#668BCC">
<h1 style="color:white">Пример: Закрываем окна</h1>
<p> </p>
<form name="form1" method="post" action="window_close_test.htm">
  <p>Для того, чтобы закрыть <b>текущее</b> окно, надо выполнить
     оператор <span class="tag_white"> self.close()</span></p>
  <p class="tag_white">
    <p> </p>
<input type="button" name="close" value="Закрыть текущее окно"
     onclick="self.close()">
  </p>
</form>
</body>
</html>
```

```
<!doctype HTML public "-//W3C//DTD HTML 4.01//EN"
<html>
   <head>
      <script type="text/javascript">
         function count_rabbits() {
            for(var i=1; i<=3; i++) {
               // оператор + з'єднує рядки
               alert("З капелюха дістали "+i+" кролика!")
            }           }
      </script>
   </head>
   <body>
      <input type="button" onclick="count_rabbits()" value="
    Рахувати кролів!"/>
   </body>
</html>
```

Об'єкт Date

- Об'єкт дозволяє працювати з датою і часом.
- створення зкземпляра
- **new Date()**
- **new Date(year, month, day)**
- **new Date(year, month, day, hours, minutes, seconds)**

Властивості об'єкту Date

параметр	**значення**	**діапазон**
year	рік	наприклад, 2000
month	місяць	0..11 (січень.. грудень)
day	день місяця	1..31
hours	годинник	0..23
minutes	хвилини	0..59
seconds	секунди	0..59

Методи об'єкту Date

метод	опис
getYear()	Повертає рік. До 2000 року повертає тільки дві останні цифри.
setYear(year)	Встановлює рік.
getMonth()	Повертає місяць.
setMonth(month)	Встановлює місяць.
getDate()	Повертає день місяця.
setDate(day)	Встановлює день місяця.
getDay()	Повертає день тижня (0-воскресенье, 1-понедельник ... 6-суббота).

getHours()	Повертає годину
setHours(hours)	Встановлює годину
getMinutes()	Повертає хвилини.
setMinutes(minutes)	Встановлює хвилини. **var d =new Date();** d.setMinutes(59);
getTime()	Повертає число мілісекунд що пройшли з 0 годин 1 січня 1970.
setTime(time)	Встановлює число мілісекунд що пройшли з 0 годин 1 січня 1970. **var d =new Date();** d.setTime(0); **var t =d.getMinutes();** alert(t + "\nYear=" + d.getYear() + "\nMonth=" + d.getMonth());

Об'єкт Array

- Представлення масивів даних і операцій над ними.
- створення зкземпляра
- **new Array()**
- **new Array(element1,element1...,elementN)**
- Параметри element1, element1 ..., elementN задають елементи масиву. Конструктор без параметрів створює порожній масив. Екземпляр також створюється за умовчанням при такому описі змінною:
- **var set = ["скольжение","причитание","уможжение","пиление"];**
- Цей запис еквівалентний наступним:
- **var set = new Array("скольжение","причитание", "уможжение","пиление");**

- Нумерація елементів масиву починається з нуля. Для доступу до окремих елементів масиву використовують конструкцію:
- імя_массива[індекс]
- Наприклад, set[0] -- це елемент "ковзання".
- Довжина масиву (число елементів в нім) може мінятися під час роботи програми.
- set[10]= "хроматика";
- alert(set);

Властивості об'єкту Array

параметр	**значення**
length	Довжина масиву (число елементів в нім). **var set= new Array(0,1,2,3,4,5,6,7,8,9,10);**

Методи об'єкту Array

метод	опис
concat(array)	Повертає масив, отриманий додаванням масиву array. Початковий масив не міняється. Приклад: **var set1 = new Array (1,2);** **var set2 = new Array (3,4);** **var set = set1.concat(set2);** alert(set1+"\n"+set2+"\n"+set); Метод concat працює тільки в браузерах, версія яких вище 3.

reverse()	Переставляє елементи в масиві так, що перший елемент стає останнім.
	var set = Array (1,2,3);
	set.reverse();
	alert(set);

slice(ind1,ind2) slice(ind1)	Повертає масив, отриманий "вирізуванням" з початкового масиву частини з позиції ind1 по позицію ind2-1. Тобто, вирізуються елементи з set[ind1] по set[ind2-1]. Початковий масив не міняється. Якщо ind2 менше нуля, то відлік останнього учасника виділення ведеться з кінця масиву. Вирізуються елементи з set[ind1] по set[set.length-ind2-1]. Якщо ind2 опущений, виділення продовжується до кінця масиву. Тобто, вирізуються елементи з set[ind1] по set[set.length-1]. Приклад: **var set = new Array (0,1,2,3);** **var set1 = set.slice(1,3);** **var set2 = set.slice(1-1);** **var set3 = set.slice(1);** alert("set=" + set + "\nset1=" + set1 + "\nset2=" + set2 + "\nset3=" + set3);Метод slice працює тільки в браузерах, версія яких вище 3.

sort(function)
sort()
	Сортування масиву. Параметром є ім'я функції, яка задає правила порівняння двох елементів. Якщо параметр опущений, елементи сортуються в лексикографічному порядку:
	var set= new Array("zebra","ant","dog","cat");
	set.sort();
	alert(set);
	Функція function повинна містити два аргументи і повертати:
	негативне число, коли перший аргумент вважається розташованим лівіше за другий в сенсі визначуваний порядок;
	0, коли аргументи вважаються рівнозначними в сенсі визначуваного порядку;
	позитивне число, коли перший аргумент вважається правішим за другий в сенсі визначуваний порядок.
	var set = new Array (26,71,9,1);
	function Compare(a,b)
	{
	return a-b;
	}
	set.sort(Compare);