Міністерство освіти і науки України
Прикарпатський національний університет імені Василя Стефаника
Кафедра інформатики

Горєлов В. О.

Конспект лекцій з дисципліни "Web-програмування" Частина 2

Івано-Франківськ, 2010

ЗМІСТ

1 Загальні відомості, вирази, базові оператори

1.1 Загальні відомості

JS – інтерпретатор з елементами об'єктно-орієнтованої моделі. Хоча він і позбавлений можливостей створення власних класів, але оперує стандартними об'єктами. Оскільки обробник знаходиться на комп'ютері користувача, JS, будучи інтерпретатором, використовує методи і властивості об'єктів браузера на комп'ютері користувача. JS надає можливість написання функцій користувача, володіє рядом операторів, працює з об'єктами, їхніми методами, властивостями і подіями. Також JS підтримує ієрархію успадкування властивостей об'єктів.

Скрипти можуть знаходитися у довільному місці `HTML`-документу. Проте теги `HTML` не можна розташовувати усередині JS-коду. JS-код знаходиться між тегами `<script> ... </script>`, виняток становлять обробники подій.

Зустрівши тег `<script>`, браузер аналізує вміст документа до тих пір, поки не буде досягнуто тегу `</script>`. Після цього проводиться перевірка скрипта на наявність помилок і компіляція JS-програми у формат, придатний для виконання на комп'ютері користувача.

Головна частина JS-програми може бути подана у контейнері `<head>... </head>`, оскільки він читається при завантаженні HTML-документа одним з перших. Теоретично, скрипт можна розташовувати у довільному місці HTML-документа, хоча краще це робити перед контейнером `<body>... </body>`, тобто у заголовкові документа. Остаточний вибір за користувачем.

Синтаксис тегу:

```
<script type="text/javascript">
[текст програми]
</script>
```

1.2 Вирази мови `JavaScript`

Вираз – це поєднання змінних, операторів та методів, що повертає певне значення.

Умовні вирази використовують для порівняння одних змінних з іншими, а також з константами або значеннями, що повертають довільні вирази. У мові JS існує оператор порівняння `?`, що має синтаксис:

```
(умовний вираз) ? оператори_1 : оператори_2
```

Якщо умовний вираз дійсний – виконуються оператори_1, інакше – оператори_2. Можна також присвоювати значення змінним. Наприклад, оператор:

```
type_time = (hour >= 12) ? "PM" : "AM"
```

присвоює рядкове значення "PM" змінній type_time, якщо значення змінної hour більше або рівне 12; інакше присвоюється "AM". Оператор ? насправді є скороченим варіантом оператора if . . . else. Попередній приклад може бути записаний так:

```
if (hour >= 12)
     type_time="PM";
else
     type_time="AM";
```

1.3 Операції присвоєння

У мові JS є декілька варіантів проведення операції присвоєння:

Оператор	Дія
=	Пряме присвоєння значення лівому операндові
+=	Додає значення лівого і правого операндів і присвоює результат лівому операндові
+	Додає значення лівого і правого операндів і присвоює результат лівому операнду
++	Збільшує (інкрементує) значення лівого операнду (правий може бути відсутнім)
-=	Віднімає значення лівого і правого операндів і присвоює результат лівому операндові
-	Віднімає значення правого від лівого операндів і присвоює результат лівому операндові
--	Зменшує (декрементує) значення лівого операнду (правий може бути відсутнім)
*	Перемножує значення лівого і правого операндів і присвоює результат лівому операндові
*=	Перемножує значення лівого і правого операндів і присвоює результат лівому операндові
/	Ділить значення лівого на значення правого операндів і присвоює результат лівому операндові
/=	Ділить значення лівого на значення правого операндів і присвоює результат лівому операндові

Так, наприклад, можна записати:

```
nval *=10;
```

тобто змінна `nval` збільшує значення в 10 разів замість:

```
nval = nval * 10.
```

1.4 Операції порівняння

Оператор	Дія
==	Рівність (дорівнює)
!=	Не дорівнює
!	Логічне заперечення
>=	Більше або дорівнює
<=	Менше або дорівнює
>	Більше
<	Менше (за можливості бажано утримуватися від застосування цього типу)

Від виразів, що мають знак "<" слід відмовлятися за можливості, оскільки даний символ може мати і інше значення у HTML-документах, наприклад:

```
if mvar <h......
```

може інтерпретуватися як початок заголовку HTML. Теги HTML в JS-програмах не дозволені.

1.5 Логічні операції

Для позначення логічної операції І у мові JS використовують два символи амперсанду (&&), а для позначення логічної операції АБО – два символи вертикальної риски (||). Ці операції застосовні виключно щодо булевих значень. Наприклад, для

```
bvar1 = true;
bvar2 = false;
bvar3 = true;
```

можна записати вираз:

```
bvar1 || bvar2
```

який поверне значення `true`, оскільки для даного виразу достатньо того, щоб значення одного з операндів було `true`. А вираз

```
bvar1 && bvar2
```

поверне, відповідно, `false`, оскільки відпрацьовується операція логічного І.

Можна записувати і складніші вирази:

```
if ((bvar1 && bvar2) || bvar3) {

     function1();
 }

 else {

     function2();
 }
```

Такий вираз слід розуміти так: "Активізувати функцію `function1()`, якщо обидві змінні `bvar1` і `bvar2` містять значення `true`, або хоча би `bvar3` містить true, інакше викликати функцію `function2` " Для даних значень буде активізована функція `function1`(), оскільки `bvar3` містить значення `true`.

1.6 Базові оператори

Програми на мові JS зазвичай складаються з програмних блоків або одиничних операторів.

Програмні блоки – це групи операторів, які обмежені фігурними дужки (`{ i }`). Кожний оператор, якщо він займає єдиний рядок, завершується крапкою з комою (`;`), що позначає закінчення оператора.

Кожний оператор має власний синтаксис. Синтаксис оператора – це набір правил, що визначають обов'язкові і допустимі для використання у даному операторі значення. Необов'язкові значення при описі синтаксису прийнято брати у квадратні дужки, наприклад `[value]`. При недотриманні правил синтаксису відбудеться помилка інтерпретації.

1.7 Оператори коментарів та приміток

Синтаксис:

```
// Текст коментарів

/*
    Текст  коментарів
*/
```

Відповідно, перший коментар може мати тільки один рядок, другий – декілька. Коментарі потрібні виключно для пояснень або для тимчасового виключення деяких фрагментів програми під час відлагодження.

1.8 Оператори циклів

Мова JS володіє рядом операторів для виконання ітерацій, тобто повторювання дії операторів.

Цикл For. Синтаксис:

```
for ([ініціалізація початкового значення;] [умова;] [механізм
     оновлення лічильника, крок]) {

      програмний блок

}
```

Оператор `for` робить можливим багатократне виконання операторів в JS-програмі. Оператор `for` може бути використаний для виконання одного або декількох операторів. Фігурні дужки можна опустити, якщо тіло циклу містить тільки один оператор. Параметри оператора `for` є необов'язковими і використовуються для управління процесом виконання циклу. При застосуванні параметрів кожну частину потрібно відокремлювати крапкою з комою (*;*).

Приклад виведення у вікні браузера таблички множення:

```
<html>
<head>
<script type="text/javascript">
<!--
function testloop() {
     document.open();
     document.writeln ('<table border="1">') ;

     for (var y = 1; y<10; y++){
          document.writeln ('<tr>');
          for (var x = 1; x<10; x++)
               document.writeln ('<td>'+x*y+'</td>');
          document.writeln ('</tr>');
     }
     document.writeln ('</table>');

     document.close();
}
//-->
</script>
</head>
<body>
<form>
<input type="button" value="Обчислити" onClick="testloop()">
</form>
</body>
</html>
```

Цикл `while`
Синтаксис:

```
while (умова) {

програмний блок
```

```
}
```

За допомогою оператора while можна виконувати один або декілька операторів до тих пір, поки не буде виконана умова. Якщо у тілі циклу виконується декілька операторів, їх необхідно обмежити фігурними дужками.

Наведемо приклад програми, яка також будує таблицю множення, але замість оператора for використовує оператор while:

```
<html>
<head>
<script type="text/javascript">
<!--
function testloop() {
     document.open();
     document.writeln ('<table border="1">') ;
     var x;
     var y=1;
     while(y<10){
          document.writeln ('<tr>');
          x=1;
          while(x<10) {
               document.writeln ('<td>'+x*y+'</td>');
               x+=1;
          }
          document.writeln ('</tr>');
          y+=1;
     }
     document.writeln ('</table>');

     document.close();
}
//-->
</script>
</head>
<body>
<form>
<input type="button" value="Обчислити" onClick="testloop()">
</form>
</body>
</html>
```

Таблиця виводиться за допомогою стандартної функції мови JS `writeln()`.

Даний приклад можна було також написати і іншим способом, використовуючи цикл for:

1.8 Вихід з циклу – оператор `break`

Синтаксис:

```
Break .
```

Оператор `break` використовується для виходу з довільного циклу, наприклад з циклу `for` або `while`. Виконання циклу припиняється в тій точці, в якій розташовано даний оператор, а керування передається наступному оператору, що знаходиться безпосередньо після циклу. Розглянемо наступну програму:

```
<html>
<head>
<script type="text/javascript">
<!--
function breaktest() {
     var index = 1;
     while (index <= 10) {
          if (index == 6)
          break;
          index ++;
     }
     document.open();
     document.writeln('цикл зупинено при index='+index);
     document.close();
}
breaktest();
//-->
</script>
</head>
</html>
```

У даному прикладі змінній `index` присвоюється значення 1, а цикл while повинен виконуватися до тих пір, поки значення змінної `index` менше або рівно 10-ти (`index` <= 10). Проте оператор `if` перевіряє виконання умови `index == 6`. Якщо ця умова виконується, то цикл `while` завершується за допомогою оператора `break`. В результаті цикл `while` завжди завершуватиметься після перших шести ітерацій, а значення змінної `index` ніколи не досягне 10-ти.

1.9 Продовження циклу – оператор `continue`

Синтаксис:

```
Continue .
```

Оператор `continue` використовують для переривання виконання блоку операторів, які складають тіло циклу і продовження циклу у наступній ітерації. На відміну від оператора `break`, оператор `continue` не зупиняє виконання циклу, а навпаки – запускає нову ітерацію. Якщо у циклі `while` цей оператор запускає нову ітерацію, то в циклах `for` відбувається оновлення змінної циклу згідно з встановленим кроком.

1.10 Визначення функції

Синтаксис:

```
function functionname (arg . . .) {

блок операторів

}
```

Функція - це блок з одного або декількох операторів. Блок виконує певні дії, а потім, можливо, повертає значення. У мові JS процедури – підпрограми що не повертають значень, не розрізняються. Всі підпрограми описуються функціями, а якщо у функцію або з неї не передаються параметри, то після імені функції ставлять круглі дужки без параметрів. Якщо функція має декілька аргументів, вони відділяються комами. Потрібно також пам'ятати, що в мові JS усередині однієї функції не може існувати іншої функції. Фігурні дужки визначають тіло функції. Функція не може бути виконана до тих пір, поки не буде явного виклику її.

Якщо необхідно, щоб функція повертала певне значення, слід використовувати необов'язковий оператор `return`, при цьому вказавши значення, яке необхідно повернути.

1.11 Повернення значення функціями - оператор `return`

Синтаксис:

```
return (value);
return value;
```

Оператор `return` завершує виконання функції і повертає значення заданого виразу. Дужки в цьому операторі можна не використовувати. Оператор `return` може бути відсутнім у функції, якщо функція не повертає значення.

Оператор `return` зазвичай використовують для повернення одного значення, проте його можна застосовувати для повернення масиву:

```
function returnarray() {
    var somearray = new Object();
    somearray[1]= "Java";
    somearray[2]= "Script";
    return (somearray);
}
```

Звертання до аргументів функції може здійснюватися за допомогою масиву `arguments[]`. У цьому масиві зберігається список аргументів, що передають поточній функції. Так, перший елемент масиву `arguments[0]` містить перший аргумент функції, `arguments[1]` - другий і т.д. Загальна кількість аргументів зберігається у властивості `arguments.length`. Приклад, який виводить на екран всі аргументи, що передають функції:

Зверніть увагу, що функцію викликають з двома аргументами, хоча у її описі задано три. В цьому випадку останній аргумент визначається як `null`. У функції `showargs()` створюється рядок аргументів, який потім виводиться за допомогою методу `alert()`.

```
<html>
<head>
<script type="text/javascript">
<!--
function showargslist(a, b, c){
     var arglist = "";
     for (var n=0; n <= arguments.length; n++) {
          arglist += n +"." + arguments[n]+ "\n";
     }
     alert(arglist);
}
showargslist("Javascipt","language")
//-->
</script>
</head>
</html>
```

1.12 Умовні оператори – `if . . . else`

Синтаксис:

```
 if (condition); {
 Програмний блок1
 } [ else { програмний блок2 }]
```

Оператор `if . . . else` – це умовний оператор, який забезпечує виконання одного або декількох операторів, залежно від того, чи задовольняються умови. Частина `condition` оператора `if` є виразом, за істинності якого виконуються оператори мови в першому програмному блоці. Програмний блок повинен бути поміщений у фігурні дужки, проте якщо використовується тільки один оператор, можна дужки не ставити. Необов'язкова частина `else` забезпечує виконання операторів другого блоку, у випадку, якщо умова condition оператора `if` є помилковою. Операторів `if` можна вкладати один в одного. Наприклад:

```
<html>
<head>
<script type="text/javascript">
<!--
today = new Date();
document.open();
minutes = today.getMinutes();
if (minutes >=0 && minutes <= 30)
     document.writeln("Це – перша півгодина");
else document.writeln("Це – перша півгодина ");
document.close();
//-->
</script>
</body>
</html>
```

1.13 Тернарний оператор ?

Синтаксис:

```
(expression) ? trueStatements ? falseStatements;
```

де `expression` – вираз на мові JS, результат виконання якого рівний або `true` (істина), або `false` (не істина). Замість `trueStatements` і `falseStatements` підставляються один або декілька операторів JS, які виконуються залежно від результату обчислення виразу `expression`. Оператори `trueStatements` виконуються, якщо вираз істинний, а `falseStatements` – якщо він помилковий. Оператор `?` можна розглядати як скорочений варіант запису оператора `if . . . else`. Наприклад:

```
<html>
<head>
<script type="text/javascript">
<!--
var today = new Date();
var secs = today.getSeconds();
(secs >=0 && secs <= 30) ?
document.write("Це – перша півгодина") :
document.write("Це – друга півгодина");
//-->
</script>
</body>
</html>
```

Вкладені оператори `?`

Для перевірки декількох умов оператори `?` можна вкладати один в одного. Наприклад:

```
<html>
<head>
<script type="text/javascript">
<!--
var today = new Date();
var minutes = today.getMinutes();
(minutes >=0 && minutes <= 30) ?
(minutes >=0 && minutes <= 15) ? document.write("Це – перша
чверть години") : document.write("Це – друга чверть години") :
document.write("Це – друга півгодина");
//-->
</script>
</body>
</html>
```

1.14 Створення змінних

Змінні створюють за допомогою оператора `var` або шляхом безпосереднього присвоєння значень за допомогою оператора (=).

Оператор: `var`.

Синтаксис:

```
var variablename [= value | expression];
```

Оператор `var` створює нову змінну з ім'ям `variablename`. Область дії цієї змінної буде або локальною, або глобальною залежно від того, де створена змінна.

Фактично, при створенні змінної оператор `var` можна опустити, проте у цьому випадку у правій частині оператора присвоєння повинно бути вказане значення, яким ініціалізують змінну.

Змінна, створена усередині функції, буде недоступна поза її межами, тобто – локальною.

1.15 Звертання до поточного об'єкту – оператор `this`

Синтаксис:

```
this[.property]
```

Оператор this є не стільки оператором, скільки внутрішньою властивістю мови JavaScript. Значення `this` є поточним об'єктом, що має стандартні властивості, такі як `name`, `length` і `value`. Оператора `this` не можна використовувати поза областю дії функції або виклику функції. Коли аргумент `property` опущений, за допомогою оператора `this` передається поточний об'єкт. Проте, при звертанні до об'єкту, як правило, потрібно вказати його конкретну властивість.

Оператор `this` застосовується для "усунення неоднозначності" об'єкту за допомогою прив'язки його до області дії поточного об'єкту, а також для зменшення кількості програмного коду.

Наприклад, можна викликати JS-функцію при обробці події `OnChange`, пов'язаної зі зміною вмісту поля для вводу даних, використовуючи оператор `this` для передачі поточного значення об'єкту:

```
<html>
<head>
<script type="text/javascript">
<!--
function upperCase(x){
     var y=document.getElementById(x).value;
     document.getElementById(x).value=y.toUpperCase();
}
//-->
</script>
</head>
<body>
Введіть ім'я:
<input type="text" id="fname" onchange="upperCase(this.id)">
```

```
</body>
</html>
```

Для тестування даного прикладу клікність після введення даних у поле де-небудь поза межами поля – введені дані повинні змінити свій регістр на верхній.

1.16 Встановлення поточного об’єкта – оператор `with`

Синтаксис:

```
with (objname) {
     statements
}
```

Оператор `with` робить об'єкт, позначений як `objname`, поточним об'єктом для операторів в програмному блоці `statements`. Зручність використання цього оператора полягає в тому, що такий запис дозволяє скоротити об’єм тексту програми. Нижче показано, як оператор with застосовується до вбудованого об'єкту Math мови JS.

```
with (Math) {
     document.writeln(PI);
}
```

2 Об’єктна модель мови. Об’єкти браузера

2.1 Об’єктна модель мови `JavaScript`

При створенні HTML-документів і JavaScript-програм необхідно враховувати структуру об’єктів. Усі об’єкти можна поділити на три групи:

а) об’єкти браузера;

б) внутрішні (вбудовані) об’єкти мови JavaScript;

в) об’єкти, пов’язані з тегами мови HTML.

Об’єктами браузера є залежні від браузера об’єкти: `window` (вікно), `location` (розташування) і `history` (історія). Внутрішні об’єкти включають прості типи даних, такі як рядки (`string`), математичні константи (`math`), дата (`date`) та інші.

Об’єкти, пов’язані з тегамі HTML, відповідають тегам, які формують поточний документ. Вони включають такі елементи як гіперзв’язки та форми.

2.2 Методи об’єктів

З об’єктами пов’язані методи, які дозволяють керувати цими об’єктами, а також у деяких випадках міняти їхній вміст. Крім того, у мові JavaScript є можливість створювати свої методи об’єктів. При використанні методу об’єкту, потрібно перед іменем методу вказати ім’я об’єкту, до якого він належить.

Наприклад, вірним викликом методу `write()` є вираз `document.write()`. А просто вираз `write()` – приведе до помилки.

2.3 Властивості об’єктів мови JavaScript

В об’єктно-орієнтованому програмуванні використовується також термін «властивість». Властивість – це іменоване значення, яке належить об'єкту. Усі стандартні об’єкти мови JS мають властивості.

Для звертання до властивості необхідно вказати імена об’єкта та властивості, розділивши їх крапкою. Кожен об’єкт має власний набір властивостей. Набір властивостей нового об'єкту можна задати при визначенні об’єкту.

Деякі властивості об’єктів доступні тільки для читання.

2.4 Об’єкти браузерів

Браузери підтримують об’єкти різних типів. HTML-об’єктами є об’єкти, які відповідають тегам мови HTML. До ним належать: мітки, гіперзв’язки і елементи форми – текстові поля, кнопки, списки і ін. Об’єкти верхнього рівня, або об’єкти браузера, – це об’єкти, підтримувані в середовищі браузера: `window`, `location`, `history`, `document`, `navigator`. Об’єкти, наведені нижче, створюються автоматично при завантаженні документа в браузер:

а) `window` – об’єкт верхнього рівня в ієрархії об’єктів мови JavaScript. Документ з фреймом також володіє об’єктом `window`;

б) `document` – містить властивості, які відносять до поточного HTML-документа, наприклад ім’я кожної форми, кольори, що використовуються для відображення документа та ін. У мові JS більшості HTML-тегів відповідають властивості об’єкта `document`;

в) location – містить властивості, що описують розташування поточного документу, наприклад адреса URL;

г) navigator – містить інформацію про версію браузера. Наприклад, властивість navigator.appname містить рядкове значення імені браузера;

д) history – містить інформацію про всі ресурси, до яких користувач звертався під час поточного сеансу роботи з браузером.

2.5 Об'єкт window

Об'єкт window звичайно відповідає головному вікну браузера і є об'єктом верхнього рівня мови JavaScript, оскільки документи, власне, і відкриваються у вікні. У документах, що містять фрейми, об'єкт window може не завжди відповідати головному вікну програми. Тому для звертання до конкретного вікна слід використовувати властивість frames об'єкта parent. Фрейми – це ті ж вікна. Щоб звернутися до них у мові JavaScript, можна скористатися масивом frames. Наприклад, вираз parent.frames[0] звертається до першого фрейма вікна браузера. Передбачається, що таке вікно існує, але за допомогою методу window.open() можна відкривати і інші вікна і звертатися до них за допомогою властивостей об'єкта window.

Для звертання до методів і властивостей об'єкта window використовують наступні варіанти запису:

```
- window.propertyName;
- window.methodName (parameters);
- self.propertyName;
- self.methodName (parameters);
- top.propertyName;
- top.methodName (parameters);
- parent.propertyName;
- parent.methodName (parameters);
- windowVar.propertyName;
- windowVar.methodName (parameters);
- propertyName;
- methodName (parameters).
```

Тут windowVar – екземпляр об'єкту window. Ім'я self - синонім, що використовують для звертання до поточного вікна у документі з фреймом, тоді як ім'я top застосовується для звертання до головного вікна браузера. З цією метою можна застосувати і об'єкт parent. Проте, слід мати на увазі, що значенням parent є посилання на батьківське вікно, тоді як top – посилання на вікно верхнього рівня, яке містить або даний фрейм. Звертання безпосередньо до методів і властивостей можливе при використанні оператора with.

Колекції об'кта window

Колекція	Опис	IE	F	O
frames[]	Повертає усі іменовані фрейми вікна	4	1	9

Властивості. Об'єкт `window` має наступні властивості:

Властивість	Опис	IE	F	O
`closed`	Повертає значення: закрито вікно чи ні	4	1	9
`defaultStatus`	Встановлює або повертає текст по замовчуванню рядка статусу вікна	4	–	9
`document`	Надає доступ до об'єкта document	4	1	9
`history`	Надає доступ до об'єкта history	4	1	9
`length`	Sets or returns the number of frames in the window	4	1	9
`location`	Надає доступ до об'єкта location	4	1	9
`name`	Встановлює або повертає ім'я вікна	4	1	9
`opener`	Повертає посилання на батьківське вікно	4	1	9
`outerHeight`	Повертає або встановлює зовнішню висоту вікна	–	1	–
`outerWidth`	Повертає або встановлює зовнішню ширину вікна	–	1	–
`parent`	Повертає батьківське вікно	4	1	9
`self`	Повертає посилання на поточне вікно	4	1	9
`status`	Встановлює текст у рядкові статусу вікна	4	–	9
`top`	Повертає прабатьківське вікно	4	1	9

Методи об'кта `window`

Метод	Опис	IE	F	O
`alert()`	Виводить повідомлення з текстом та OK кнопкою	4	1	9
`blur()`	Видаляє фокус з поточного вікна	4	1	9
`clearInterval()`	Відміняє таймаут, встановлений з setInterval()	4	1	9
`clearTimeout()`	Відміняє таймаут, встановлений з setTimeout()	4	1	9
`close()`	Закриває поточне вікно	4	1	9
`confirm()`	Відображає діалогове вікно з повідомленням і кнопками OK та Cancel	4	1	9
`createPopup()`	Створює pop-up-вікно	4	–	–
`focus()`	Встановлює фокус на поточному вікні	4	1	9
`moveBy()`	Зміщує вікно відносно його поточного положення	4	1	9
`moveTo()`	Переміщує вікно у вказану позицію	4	1	9
`open()`	Відкриває нове вікно браузера	4	1	9
`print()`	Друкує вміст поточного вікна	5	1	9
`prompt()`	Виводить діалогове вікно з підказкою для введення інформації	4	1	9
`resizeBy()`	Змінює розміри вікна на вказану у пікселях величину	4	1	9
`resizeTo()`	Змінює розміри вікна до вказаних ширини та висоти	4	1.5	9
`scrollBy()`	Прокручує вміст вікна на вказану кількість пікселів	4	1	9
`scrollTo()`	Прокручує вміст вікна до вказаної позиції у пікселяї	4	1	9
`setInterval()`	Циклічно виконує вираз через вказані інтервали часу	4	1	9
`setTimeout()`	Одноразово виконує вираз через вказаний інтервал у мс	4	1	9

Метод `alert()` застосовують для виведення на екран текстового повідомлення. Для відкриття вікна використовують метод `open()`, а для закриття – метод `close()`. За допомогою методу `confirm()` відбувається виведення на екран вікна повідомлення з кнопками Yes і No, метод повертає булеве значення `true` або `false` залежно від натиснутої

кнопки. За допомогою методу `prompt()` на екран виводять діалогове вікно з полем введення. Метод `setTimeout()` встановлює в поточному вікні обробку подій, пов'язаних з таймером, а метод `clearTimeout()` відміняє обробку таких подій.

Обробники подій

Об'єкт `window` не обробляє події до тих пір, поки у вікно не завантажений документ. Проте, можна обробляти події, пов'язані із завантаженням і вивантаженням документів. Обробники таких подій задаються як значення атрибутів `onLoad` і `onUnload`, що вказуються у тегові `<body>`. Ці ж атрибути можуть бути визначені в тегах `<frameset>` документів, що містять фрейми.

2.6 Об'єкт `document`

Об'єкт `document` відповідає цілому гіпертекстовому документлві, а точніше, тій його частини, яка знаходиться у контейнері `<body> . . . </body>`. Документи відображаються у вікнах браузера, тому кожний із них пов'язаний з певним вікном. Усі HTML-об'єкти є властивостями об'єкту `document`, тому вони знаходяться у самому документі. Наприклад, на мові JS до першої форми документа можна звернутися, використовуючи вираз:

```
document.forms[0] ,
```

тоді як до першої форми в другому фреймі слід звертатися виразом:

```
parent.frames[1].document.forms[0].
```

Об'єкт `document` зручно використовувати для динамічного створення HTML-документів. Для цього застосовується HTML-контейнер `<body> . . . </body>`. Хоча в цьому контейнері можна встановити різноманітні властивості документа, все ж є такі властивості, значення яких не можна встановити за допомогою цих тегів.

Для звертання до властивостей і методів об'єкту `document` застосовується наступний синтаксис:

```
document.propertyName
document.methodName (parameters)
```

Колекції об'єкта `document`

Колекція	Опис	IE	F	O	W3C
`anchors[]`	Повертає посилання на усі об'єкти Anchor документа	4	1	9	Yes
`forms[]`	Повертає посилання на усі об'єкти Form документа	4	1	9	Yes
`images[]`	Повертає посилання на усі об'єкти Image документа	4	1	9	Yes
`links[]`	Повертає посилання на усі об'єкти Area та Link документа	4	1	9	Yes

Властивості об’єкта `document`

Властивість	Опис	IE	F	O	W3C
`body`	Надає прямий доступ до <body> елемента				
`cookie`	Встановлює або повертає куки, асоційовані з даним документом	4	1	9	Yes
`domain`	Повертає доменне ім’я поточного документа	4	1	9	Yes
`lastModified`	Повертає дату та час останньої модифікації документа	4	1	No	No
`Referrer`	Повертає URL документа, що завантажив поточний документ	4	1	9	Yes
`title`	Повертає title поточного документа	4	1	9	Yes
`URL`	Повертає URL поточного документа	4	1	9	Yes

Методи об’єкта `document`

Method	Description	IE	F	O	W3C
`close()`	Закриває потік виведення, відкритий методом document.open() method, і виводить зібрані дані	4	1	9	Yes
`getElementById()`	Повертає посилання на перший об’єкт із вказаним id	5	1	9	Yes
`getElementsByName()`	Повертає колекцію об’єктів із вказаним іменем	5	1	9	Yes
`getElementsByTagName()`	Повертає колекцію об’єктів із вказаним тегом	5	1	9	Yes
`open()`	Відкриває потік для збору даних для виведення з document.write() чи document.writeln() методами	4	1	9	Yes
`write()`	Записує HTML вирази чи JavaScript-код у документ	4	1	9	Yes
`writeln()`	Ідентичний методові write() за винятком того, що додає у кінці виразу символ початку нового рядка	4	1	9	Yes

Обробники подій

У тегах `<body>` і `<frame>` можна використовувати обробники подій, пов’язаних із завантаженням та вивантаженням документа, `onLoad` і `onUnload`.

За допомогою методу write() можна динамічно створювати теги зображень, виводячи зображення на екран таким чином:

```
document.open();
document.writeln("<img src='myimage.gif'>");
document.close();
```

За допомогою JavaScript програм, а зокрема за допомогою об’єкту `document`, можна створювати закінчені HTML-документи та інші JavaScript програми.

```
<html>
<head>
<script type="text/javascript">
     document.open();
     document.writeln("<html><body>
          <table
     border=1><tr><td>test</td></tr></table></body></html>");
     document.close();
</script>
</head>
<body>
</body>
</html>
```

2.7 Об'єкт location

Даний об'єкт зберігає інформацію про розташування поточного документа у вигляді його URL. При управлінні об'єктом location існує можливість змінювати адресу URL документа. Об'єкт location пов'язаний з поточним об'єктом window – вікном, в яке завантажений документ. Документи не містять інформації про адреси URL. Ці адреси є властивістю об'єктів window.

Синтаксис:

```
windowVar.]location.prepertyName
```

де windowVar – необов'язкова змінна, що вказує на конкретне вікно, до якого звертаються. Ця змінна також дозволяє звертатися до фрейма за допомогою властивості parent – синоніма, який використовують при звертанні до об'єкту window верхнього рівня, якщо вікон декілька. Об'єкт location є властивістю об'єкту window. Якщо звертаєтеся до об'єкту location, не вказуючи імені вікна, то мають на увазі властивість поточного вікна.

Властивість location об'єкту window легко переплутати з властивістю location об'єкту document. Значення властивості document.location змінити не можна, а значення властивості location вікна – можна, наприклад за допомогою виразу window.location.property. Значення document.location присвоюється об'єктові window.location при первинному завантаженні документа, тому, що документи завжди завантажуються у вікна.

Властивості. Об'єкт location має наступні властивості:

Властивість	Опис	IE	F	O
hash	Встановлює або повертає URL з хеш-знаку (#)	4	1	9
host	Встановлює або повертає ім'я хоста та номер порту поточного URL	4	1	9
hostname	Встановлює або повертає ім'я хоста поточного URL	4	1	9
href	Встановлює або повертає повний URL	4	1	9
pathname	Встановлює або повертає шлях поточного URL	4	1	9
port	Встановлює або повертає номер порту поточного URL	4	1	9
protocol	Встановлює або повертає протокол поточного URL	4	1	9

Властивість	Опис	IE	F	O
search	Встановлює або повертає URL із помітки запиту (?)	4	1	9

Методи об'єкту location

Для об'єкту location методи не визначені, також він не пов'язаний з якими-небудь обробниками подій.

Метод	Опис	IE	F	O
assign()	Завантажує новий документ	4	1	9
reload()	Перезавантажує поточний документ	4	1	9
replace()	Замінює поточний документ на новий	4	1	9

```
<html>
<head>
<script type="text/javascript">
     self.location="http://gmail.com";
</script>
</head>
<body>
</body>
</html>
```

Даний приклад завантажує у поточне вікно Web-сторінку. Об'єкт self можна опустити, оскільки він є посиланням на поточне вікно.

2.8 Об'єкт history

Об'єкт history містить список адрес URL, відвіданих у поточному сеансі. Об'єкт history пов'язаний з поточним документом.

Синтаксис:

```
history.propertyName
history.methodName (parameters)
```

Властивості. Значенням властивості length є кількість елементів у спискові об'єкта history.

Методи. Метод back() дозволяє завантажувати в браузер попередній ресурс, тоді як метод forward() забезпечує звертання до наступного ресурсу у спискові. За допомогою методу go() можна звернутися до ресурсу з певним номером у спискові об'єкту history. Обробники подій для об'єктів history не визначені.

Властивість	Опис	**IE**	**F**	**O**
Length	Повертає число елементів у спискові history	4	1	9

Методи об'єкта history

Метод	Опис	**IE**	**F**	**O**
`back()`	Завантажує попередній URL у спискові history	4	1	9
`Forward()`	Завантажує наступний URL у спискові history	4	1	9
`go()`	Завантажує конкретну сторінку у спискові історії	4	1	9

Наведемо кілька прикладів використання об'єкта history.

Для того, щоб переглянути попередній завантажений документ, можна скористатися оператором:

```
history.go(-1);
```

або

```
history.back();
```

Для звертання до історії конкретного вікна або фрейму застосовують об'єкт `parent`:

```
parent.frames[0].history.forward();
```

завантажує в перший фрейм попередній документ. Якщо відкрито декілька вікон браузера – можна записати:

```
window1.frames[0].history.forward();
```

тут у перший фрейм вікна `window1` буде завантажено наступний документ зі списку об'єкта `history`.

2.9 Об'єкт `navigator`

Об'єкт `navigator` містить інформацію про версію поточного браузера. Цей об'єкт застосовується для отримання інформації про версії. Синтаксис:

```
navigator.propertyName
```

Методи і події не визначені для цього об'єкту. Властивості – призначені тільки для читання, оскільки ресурс з інформацією про версію недоступний для редагування.

Властивості об'єкта `navigator`

Властивість	Опис	IE	F	O
`appCodeName`	Повертає кодове ім'я браузера	4	1	9
`appMinorVersion`	Повертає номер підверсії браузера	4	No	No
`appName`	Повертає ім'я браузера	4	1	9
`appVersion`	Повертає та версію браузера	4	1	9
`browserLanguage`	Повертає поточну мову браузера	4	No	9
`cookieEnabled`	Повертає булеве значення про дозвіл cookie браузером	4	1	9
`cpuClass`	Повертає значення класу процесора системи	4	No	No
`onLine`	Повертає булеве значення, яке відповідає оффлайн режимові роботи системи	4	No	No
`platform`	Повертає значення операційної системи (платформа)	4	1	9

Властивість	Опис	IE	F	O
`systemLanguage`	Повертає мову операційної системи по замовчуванню	4	No	No
`userAgent`	Повертає значеннятhe user-agent-заголовку, що надсилається клієнтом на сервер	4	1	9
`userLanguage`	Повертає значення мови операційної системи	4	No	9

Методи об’єкта `navigator`

Method	Description	IE	F	O
`javaEnabled()`	Встановлює дозвіл підтримки браузером Java	4	1	9
`taintEnabled()`	Встановлює, чи дозволено руйнування даних	4	1	9

3 Внутрішні об’єкти

3.1 Об’єкт `array`

`Array` – це багатовимірна впорядкована множина об’єктів; звертання до елементів масиву ведеться за допомогою цілочисельного індексу. Прикладами об’єктів-масивів у браузері є гіперзв’язки, мітки, форми, фрейми. Масив можна створити одним з наступних способів:

- використовуючи визначену користувачем функцію для привласнення об'єкту багатьох значень;
- використовуючи конструктор Array();
- використовуючи конструктор Object().

Властивості об’єкту `Array`

Властивість	Опис	**FF**	**N**	**IE**
`constructor`	Повертає посилання на функцію, що створила об’єкт масиву	1	2	4
`Index`		1	3	4
`input`		1	3	4
`length`	Встановлює або повертає число елементів у масиві	1	2	4
`prototype`	Надає можливість додати властивості та методи об’єкту	1	2	4

Методи об’єкту `Array`

Метод	Опис	**FF**	**N**	**IE**
`concat()`	Об’єднує два або більше масивів та повертає результат	1	4	4
`join()`	Розташовує елементи масиву у рядкові. Елементи розділяються вказаним символом-роздільником	1	3	4
`pop()`	Видаляє та повертає останній елемент масиву	1	4	5.5
`push()`	Додає один або більше елементів у кінець масиву та повертає результат	1	4	5.5
`reverse()`	Змінює порядок розташування елементів масиву на протилежний	1	3	4
`shift()`	Видаляє та повертає перший елемент масиву	1	4	5.5
`slice()`	Повертає вибрані елементи із існуючого масиву	1	4	4
`sort()`	Сортує елементи масиву	1	3	4
`splice()`	Видаляє та додає нові елементи до масиву	1	4	5.5
`toSource()`	Надає код об’єкту	1	4	-
`toString()`	Перетворює довільний масив у рядок та повертає результат	1	3	4
`unshift()`	Додає один або більше елементів на початок масиву та повертає нову довжину	1	4	6
`valueOf()`	Повертає первинне значення об’єкту Array. Метод викликається JavaScript автоматично і у коді прогами неявний	1	2	4

Приклад використання масиву.

```
<html>
<body>
<script type="text/javascript">
     var mycars = new Array();
     mycars[0] = "Saab";
     mycars[1] = "Volvo";
     mycars[2] = "BMW";
     for (i=0;i<mycars.length;i++)
     {
     document.write(mycars[i] + "<br />");
     }
</script>
</body>
</html>
```

3.2 Об'єкт Date

Об'єкт містить інформацію про дату і час. Даний об'єкт має методи, призначені для отримання такого типу інформації. Крім цього, об'єкти Date можна створювати і змінювати, наприклад шляхом складання або віднімання значень дат, одержувати нову дату. Для створення об'єкту Date застосовується синтаксис:

```
dateObj = new Date(parameters),
```

де dateObj - змінна, у яку буде записаний новий об'єкт Date. Аргумент `parameters` може набувати наступних значень:

– порожній параметр, наприклад date(): в даному випадку дата і час – системні;

– рядок, що представляє дату і час у вигляді: "місяць, день, рік, час", наприклад "March 1, 2000, 17:00:00". Годину представлено у 24-годинному форматі;

– значення року, місяця, дня, години, хвилин, секунд. Наприклад, рядок "00,4,1,12,30,0" означає 1 квітня 2000 року, 12:30.

– цілі значення тільки для року, місяця і дня, наприклад "00,5,1" означає 1 травня 2000 року, одразу після опівночі, оскільки значення часу рівні нулю.

Властивості об'єкту `Date`

Властивість	Опис	**FF**	**N**	**IE**
`constructor`	Повертає посилання на Date-функцію, що створила об'єкт	1	4	4
`prototype`	Надає можливість додавати властивості та методи до об'єкта	1	3	4

Методи об'єкту `Date`

Метод	Опис	**FF**	**N**	**IE**
`Date()`	Повертає поточну дату та час	1	2	3
`getDate()`	Повертає значення дня місяця об'єкту Date (1-31)	1	2	3
`getDay()`	Повертає значення дня тижня об'єкту Date (0-6)	1	2	3

Метод	Опис	FF	N	IE
`getFullYear()`	Повертає значення року об'єкту Date (4 цифри)	1	4	4
`getHours()`	Повертає значення години об'єкту Date (0-23)	1	2	3
`getMilliseconds()`	Повертає значення у мілісекундах об'єкту Date (0-999)	1	4	4
`getMinutes()`	Повертає значення хвилин об'єкту Date (0-59)	1	2	3
`getMonth()`	Повертає значення місяця об'єкту Date (0-11)	1	2	3
`getSeconds()`	Повертає значення секунд об'єкту Date (from 0-59)	1	2	3
`getTime()`	Повертає кількість мілісекунд, що сплили з 1січня 1970р.	1	2	3
`getTimezoneOffset()`	Повертає різницю у хвилинах між локальним часом та Greenwich Mean Time (GMT)	1	2	3
`getUTCDate()`	Повертає значення дня місяця згідно із UT (1-31)	1	4	4
`getUTCDay()`	Повертає значення дня тижня згідно із UT (0-6)	1	4	4
`getUTCMonth()`	Повертає значення місяця згідно із UT (0-11)	1	4	4
`getUTCFullYear()`	Повертає 4 цифри року згідно із UT	1	4	4
`getUTCHours()`	Повертає значення годин об'єкту Date згідно із UT (0-23)	1	4	4
`getUTCMinutes()`	Повертає значення хвилин об'єкту Date згідно із UT (0-59)	1	4	4
`getUTCSeconds()`	Повертає значення секунд об'єкту Date згідно із UT (0-59)	1	4	4
`getUTCMilliseconds()`	Повертає значення мілісекунд об'єкту Date згідно із UT (from 0-999)	1	4	4
`getYear()`	Повертає значення року як дво або три/чотири-символьне число залежно від браузера. Варто використовати getFullYear() замість цього методу	1	2	3
`parse()`	Приймає як параметр рядок із датою і повертає кількість мілісекунд, що сплили, починаючи від опівночі 1.01.1970	1	2	3
`setDate()`	Встановлює день місяця (1-31)	1	2	3
`setFullYear()`	Встановлює рік (four digits)	1	4	4
`setHours()`	Встановлює годину (from 0-23)	1	2	3
`setMilliseconds()`	Встановлює мілісекунди (from 0-999)	1	4	4
`setMinutes()`	Встановлює хвилини (from 0-59)	1	2	3
`setMonth()`	Встановлює місяці (from 0-11)	1	2	3
`setSeconds()`	Встановлює секунди (from 0-59)	1	2	3
`setTime()`	Встновлює дату та час шляхом додавання чи віднімання вкачаного числа мілісекунд до/від опівночі 1.01.1970	1	2	3
`setUTCDate()`	Встановлює день місяця згідно із UT (1-31)	1	4	4
`setUTCMonth()`	Встановлює місяць згідно із UT (0-11)	1	4	4
`setUTCFullYear()`	Встановлює рік у форматі UT (4 цифри)	1	4	4
`setUTCHours()`	Встановлює годину згідно із UT (0-23)	1	4	4
`setUTCMinutes()`	Встановлює хвилини згідно із UT(0-59)	1	4	4
`setUTCSeconds()`	Встановлює секунди згідно із UT (from 0-59)	1	4	4
`setUTCMilliseconds()`	Встановлює мілісекунди згідно із UT (from 0-999)	1	4	4
setYear()	Встановлює рік (2 або 4 цифри). Варто використовувати setFullYear() замість цього методу	1	2	3

Метод	Опис	FF	N	IE
`toDateString()`	Повертає значення об'єкту у читабельній формі			
`toGMTString()`	Конвертує об'єкт у відповідності до Greenwich time у рядок. Використовуйте toUTCString() замість цього методу	1	2	3
`toLocaleDateString()`	Конвертує значення об'єкта у локальному форматі у рядок і повертає його (дата)	1	4	4
`toLocaleTimeString()`	Конвертує значення об'єкта у локальному форматі у рядок і повертає його (час)	1	4	4
`toLocaleString()`	Конвертує значення об'єкта у локальному форматі у рядок і повертає його	1	2	3
`toSource()`	Надає вихідний код об'єкту	1	4	-
`toString()`	Конвертує об'єкт у рядок і повертає його	1	2	4
`toTimeString()`	Повертає час у читабельній формі			
`toUTCString()`	Конвертує об'єкт згідно із UT	1	4	4
`UTC()`	Приймає як параметр дату і повертає кількість мілісекунд, що сприли з опівночі 1.01 1970 згідно з UT	1	2	3
`valueOf()`	Повертає первинне значення об'єкта Date. Метод викликається JavaScript автоматично і у коді прогами неявний	1	2	4

3.3 Об'єкт `Math`

Об'єкт Math – вбудованим об'єкт мови JavaScript, що містить властивості і методи, які використовують для виконання математичних операцій. Об'єкт Math включає також деякі широко вживані математичні константи. Синтаксис:

```
Math.propertyName

Math.methodName(parameters) .
```

Методи об'єкту `Math`

Властивість	Опис	FF	N	IE
`E`	Повертає значення константи E (~ 2.718)	1	2	3
`LN2`	Повертає натуральний логарифм 2 (~ 0.693)	1	2	3
`LN10`	Повертає натуральний логарифм 10 (~ 2.302)	1	2	3
`LOG2E`	Повертає логарифм E при основі 2 (~1.442)	1	2	3
`LOG10E`	Повертає логарифм E при основі 10 (~ 0.434)	1	2	3
`PI`	Повертає число PI (~ 3.14159)	1	2	3
`SQRT1_2`	Повертає корінь квадратний від 1/2 (~0.707)	1	2	3
`SQRT2`	Повертає корінь квадратний від 2 (~ 1.414)	1	2	3

Властивості об'єкту `Math`

Метод	Опис	FF	N	IE
`abs(x)`	Повертає абсолютне значення	1	2	3

Метод	Опис	**FF**	**N**	**IE**
`acos(x)`	Повертає arcos	1	2	3
`asin(x)`	Повертає arcsin	1	2	3
`atan(x)`	Повертає arctan x як числове значення між -PI/2 та PI/2 радіан	1	2	3
`atan2(y,x)`	Повертає кут тета точки з координатами (x,y) як числове знаяення між -PI та PI радіан	1	2	3
`ceil(x)`	Повертає значення числа, заокруглене до найближчого більшого цілого	1	2	3
`cos(x)`	Повертає cos	1	2	3
`exp(x)`	Повертає значення E^x	1	2	3
`floor(x)`	Повертає значення числа, заокруглене до найближчого меншого цілого	1	2	3
`log(x)`	Повертає натуральний логарифм при основі E	1	2	3
`max(x,y)`	Знаходить максимальне значення з двох елементів	1	2	3
`min(x,y)`	Знаходить мінімальне значення з двох елементів	1	2	3
`pow(x,y)`	Повертає значення x^y	1	2	3
`random()`	Повертає випадкове значення між 0 та1	1	2	3
`round(x)`	Заокруглює значення до найближчого цілого	1	2	3
`sin(x)`	Повертає sin числа	1	2	3
`sqrt(x)`	Повертає значення кореню квадратного	1	2	3
`tan(x)`	Повертає значення tan кута	1	2	3
`toSource()`	Надає вихідний код об’єкту	1	4	-
`valueOf()`	Повертає первинне значення об’єкту. Метод викликається JavaScript автоматично і у коді програми неявний	1	2	4

3.4 Об’єкт `String`

Рядок (`string`) на мові JavaScript – це послідовність символів у подвійних або одинарних лапках. Для керування рядковими об’єктами використовується синтаксис:

```
stringName.propertyName ,

stringName.methodName(parameters) .
```

Тут `stringName` – ім’я об'єкту `String`. Рядки можна створювати трьома способами:

– створити рядкову змінну за допомогою оператора `var` і присвоїти їй рядкове значення значення;

– присвоїти значення рядкової змінної тільки за допомогою оператора привласнення (=);

– скориставшись конструктором `String()`.

Властивості об’єкту `String`

Властивість	Опис	**FF**	**N**	**IE**
`constructor`	Посилання на функцію, що створила об’єкт	1	4	4
`length`	Повертає кількість символів у рядкові	1	2	3
`prototype`	Надає можливість додавати властивості та методи до о’єкту	1	2	4

Методи об’єкту `String`

Метод	Опис	**FF**	**N**	**IE**
`anchor()`	Створює HTML якір	1	2	3
`big()`	Виводить рядок великим шрифтом	1	2	3
`blink()`	Виводить блимаючий рядок	1	2	
`bold()`	Виводить рядок у bold	1	2	3
`charAt()`	Повертає символ зі вказаної позиції рядка	1	2	3
`charCodeAt()`	Повертає Unicode символу у вказаній позиції	1	4	4
`concat()`	Об’єднує два і більше рядків	1	4	4
`fixed()`	Виводить рядок шрифтом фіксованої ширини (teletype)	1	2	3
`fontcolor()`	Виводить рядок вказаним кольором	1	2	3
`fontsize()`	Виводить рядок вказаним розміром	1	2	3
`fromCharCode()`	Приймає як парамер Unicode-значення і повертає символ	1	4	4
`indexOf()`	Повертає позицію першого входження вказаного підрядка у рядок	1	2	3
`italics()`	Виводить рядок italic шрифтом	1	2	3
`lastIndexOf()`	Повертає позицію першого входження вказаного підрядка у рядок, шукаючи назад від вказаної позиції у рядкові	1	2	3
`link()`	Воводить рядок як гіперпосилання	1	2	3
`match()`	Шукає вказану послідовність символів у рядкові	1	4	4
`replace()`	Замінює одні символи у рядкові іншими	1	4	4
`search()`	Шукає вказану послідовність символів у рядкові. Метод підтримує регулярні вирази	1	4	4
`slice()`	Повертає частину рядка як новий рядок	1	4	4
`small()`	Виводить рядок малим шрифтом	1	2	3
`split()`	Ділить рядок на послідовність символів і записує її у масив	1	4	4
`strike()`	Виводить рядок перекресленим шрифтом	1	2	3
`sub()`	Виводить рядок як нижній індекс	1	2	3
`substr()`	Повертає вказану кількість символів, починаючи із заданого	1	4	4
`substring()`	Повертає символи із рядка між двома вказаними індексами	1	2	3
`sup()`	Виводить рядок як верхній індекс	1	2	3
`toLowerCase()`	Виводить рядок нижнім регістром	1	2	3
`toUpperCase()`	Виводить рядок верхнім регістром	1	2	3
`toSource()`	Надає вихідний код об’єкту	1	4	-
`valueOf()`	Повертає первинне значення об’єкту. Метод викликається JavaScript автоматично і у коді програми неявний	1	2	4

4 Об’єкти, що відповідають тегам HTML

Чимало об’єктів мови JavaScript відповідають тегам, які формують HTML-документи. Кожен такий об’єкт перебуває у внутрішній ієрархії, оскільки всі вони мають загальний батьківський об’єкт – браузер, який представлений об’єктом `window`.

Деякі об’єкти мови JavaScript мають нащадків. Зокрема, гіперпосилання є об’єктом, успадкованим з об’єкту `document`. У мові JS успадковані об’єкти також називають властивостями. Наприклад, множина гіперпосилань є властивістю об’єкта `document`, а `links` – ім’ям цієї властивості. Гіперпосилання, будучи об’єктом, у той же час є властивістю `links`-об’єкту `document`.

Розглянемо HTML-приклад.

```
<html>
<head>
</head>
<body bgcolor="White">
<form>
<input type="checkbox" checked name="chck1">Item 1
</form>
</body>
</html>
```

Еквівалентний запис на JavaScript:

```
document.bgColor="White"
document.forms[0].chck1.defaultChecked=true
```

Як видно із прикладу, кольорові фону документа, встановленому тегом <body>, відповідає властивість `document.bgColor`. Перемикачеві `checkbox` з ім’ям `chck1`, визначеному у формі, відповідає вираз `document.forms[0].chck1`. Властивість `defaultChecked` належить об’єктові `check-box` і може приймати значення `true` або `false` залежно від того, чи вказаний у тегові <input> атрибут `checked`. Коли цей атрибут заданий, перемикач відображається на екрані як увімкнутий по замовчуванню.

Оскільки документ може включати декілька таких об’єктів як гіперпосилання, форми та інші об’єкти, багато хто з них є масивами. Для звертання до певного елементу масиву потрібно вказати його індекс. Наприклад, `forms[0]` – перша форма поточного документу. До другої форми можна звернутися, відповідно, через `forms[1]`.

4.1 Об’єкт `anchor`

`Anchor` – це елемент тексту, який є об’єктом призначення для тега гіперпосилання <a href>, а також властивістю об’єкта `document`. Тегові <a name> у мові JavaScript відповідає об’єкт `anchor`, а всім тегам <a name>, наявним в документі, – масив `anchors`. Будучи об’єктами призначеними для гіперпосилань, мітки в основному використовуються для індексування вмісту гіпертекстових документів, забезпечуючи швидке переміщення до певної частини документа. Тег <a>, що встановлює задаючі мітки і гіперзв'язки, має синтаксис:

```
<a [href=location]
[name="anchorName"]
[target="windowName"] >
anchorText
</a>
```

Властивості об’єкта `Anchor`

Властивість	Опис	IE	F	O	W3C
`accessKey`	Встановлює або повертає значення кнопки клавіатури (Alt+…) для доступу до гіперпосилання	5	1	No	Yes
`charset`	Встановлює або повертає набір символів (таблицю символів) для прилінкованого ресурсу	6	1	9	Yes
`coords`	Встановлює або повертає розділений комами список з координатами лінків в image-map	6	1	9	Yes
`href`	Встановлює або повертає URL зв’язаного ресурсу	5	1	9	Yes
`hreflang`	Встановлює або повертає код мови зв’язаного ресурсу	6	1	9	Yes
`id`	Встановлює або повертає id гіперпосилання	4	1	9	Yes
`innerHTML`	Встановлює або повертає текст гіперпосилання	4	1	9	No
`name`	Встановлює або повертає ім’я гіперпосилання	4	1	9	Yes
`rel`	Встановлює або повертає взаємозв’язок між поточним документом та цільовим URL	5	1	No	Yes
`rev`	Встановлює або повертає взаємозв’язок між URL поточним документом	5	1	No	Yes
`shape`	Встановлює або повертає форму лінка у межах image-map	6	1	9	Yes
`tabIndex`	Встановлює або повертає tab-порядок гіперпосилань	6	1	9	Yes
`target`	Встановлює або повертає значення, що вказує, де відкрити лінк	5	1	9	Yes
`type`	Встановлює або повертає MIME-тип прилінкованого ресурсу	6	1	9	Yes

Стандартні властивості

Властивість	Опис	IE	F	O	W3C
`className`	Встановлює або повертає значення атрибуту класу елемента	5	1	9	Yes
`dir`	Встановлює або повертає напрям тексту	5	1	9	Yes
`lang`	Встановлює або повертає значення коду мови елемента	5	1	9	Yes
`title`	Встановлює або повертає значення title елемента	5	1	9	Yes

Методи об’єкту `Anchor`

Методи	Опис	IE	F	O	W3C
`blur()`	Прибирає фокус із гіперпосилання	5	1	9	Yes
`focus()`	Встановлює фокус на гіперпосиланні	5	1	9	Yes

4.2 Масив anchors

За допомогою масиву anchors програма на мові JavaScript може звертатися до мітки поточного гіпертекстового документу. Кожному тегові <a name> поточного документу відповідає елемент масиву anchors. Для того, щоб програма виконувалася правильно, у відповідних атрибутах name повинні бути задані імена усіх міток. Якщо документ містить іменовану мітку, визначену HTML-тегом

```
<a name="s1">Selection1</a>,
```

то цій мітці в JS-програмі відповідає об'єкт document.anchors[0]. Щоб перейти до цієї мітки за допомогою гіперпосилання, користувач повинен клацнути мишею на тексті, визначеному у контейнері . . . . До масиву anchors можна звертатися за допомогою наступних операторів:

```
document.anchors[i],

document.anchors.length,
```

де i – індекс мітки. Властивість length дозволяє визначити кількість міток у документі, хоча елементи, що відповідають окремим міткам, міститимуть значення null. Це означає, що не можна звертатися до імен окремих міток через елементи масиву.

Властивості. Масив anchors має тільки одну властивість length, яка повертає значення, відповідне кількості міток в документі. Масив anchors є структурою тільки для читання.

Методів і обробників подій об'єкти anchors не мають.

```
<html>
<body>
<a name="first">Перший anchor</a><br />
<a name="second">Другий anchor</a><br />
<a name="third">Третій anchor</a><br />
<br />
Кількість об'єктів anchor документа:
<script type="text/javascript">
document.write(document.anchors.length)
</script>
</body>
</html>
```

4.3 Об'єкт button

Властивості об'єкта Button

Кнопка – це область вікна, яка реагує на клацання миші і може активізувати оператора або функцію мови JavaScript за допомогою атрибуту події onClick. Кнопки є властивостями об'єкту form і повинні знаходитися у тегах <form> . . . </form> мови HTML.

Синтаксис:

```
<input type="button"
name="buttonName"
value="buttonText"
[onClick="handlerText"]>
```

Атрибут `name` задає ім'я кнопки і у мові JS йому відповідає властивість `name` нового об'єкту `button`. Атрибут `value` визначає напис на кнопці, котрій відповідає властивість `value`. До властивостей і методів об'єкту `button` можна звернутися одним із способів:

- `buttonName.propertyName;`
- `buttonName.methodName (parameters);`
- `formName.elements[i].propertyName;`
- `formName.elements[i].methodName (parameters).`

Тут `buttonName` – значення атрибуту `name`, а `formName` - або значення атрибуту `name` об'єкту `form`, або елемент масиву `forms`. Змінна і є індексом, використовуваним для звернення до окремого елементу масиву, в даному випадку до елементу `button`.

Властивість	Опис	IE	F	O	W3C
`accessKey`	Встановлює або повертає «гарячу клавішу» для доступу до кнопки	5	1	9	Yes
`alt`	Встановлює або повертає текст для браузерів, які не підтримують кнопки	5	1	9	Yes
`disabled`	Встановлює або повертає статус кнопки щодо деактивованості	5	1	9	Yes
`form`	Повертає посилання на форму, що містить кнопку	4	1	9	Yes
`id`	Встановлює або повертає id кнопки	4	1	9	Yes
`name`	Встановлює або повертає ім'я кнопки	4	1	9	Yes
`tabIndex`	Встановлює або повертає tab-порядок кнопок	5	1	9	Yes
`type`	Повертає тип форми, елементом якої є кнопка	4	1	9	Yes
`value`	Встановлює текст, що виводиться на кнопці	4	1	9	Yes

Стандартні властивості

Властивість	Опис	IE	F	O	W3C
`className`	Встановлює або повертає атрибут класу елмента	5	1	9	Yes
`dir`	Встановлює або повертає напрям тексту на кнопці	5	1	9	Yes
`lang`	Встановлює або повертає код мови елемента	5	1	9	Yes
`title`	Встановлює або повертає title елемента	5	1	9	Yes

Методи об'єкта `Button`

Опис	Опис	IE	F	O	W3C
`blur()`	Видаляє фокус із кнопки	4	1	9	Yes
`click()`	Симулює клік миші на кнопці	4	1	9	Yes
`focus()`	Встановлює фокус на кнопці	4	1	9	Yes

4.4 Об'єкт checkbox

Контрольний перемикач – це кнопка (прапорець), яку можна встановити в один з двох станів: увімкнуто або вимкнуто. Об'єкти checkbox – це властивості об'єкту form і повинні бути розташовані у тегах <form> . . . </form>. Простий контрольний перемикач:

Синтаксис:

```
<input name="checkboxName"
type="checkbox"
value="checkboxValue"
[checked]
[onClick="handlerText"]>textToDisplay
```

де атрибут name є ім'ям об'єкту checkbox. Йому відповідає властивість name об'єкту мови JavaScript. Атрибут value визначає значення, яке передається серверові при пересиланні значень елементів форми, якщо контрольний перемикач увімкнутий. Необов'язковий атрибут checked вказує на те, що контрольний перемикач повинен бути увімкнутий по замовчуванню. Якщо цей атрибут заданий, властивість defaultChecked має значення true. За допомогою властивості checked можна визначити, чи увімкнутий контрольний перемикач. Текст, що відображається поряд з контрольним перемикачем, задається рядком textToDisplay.

До об'єкту checkbox можна звертатися одним із способів:

- checkboxName.propertyName
- checkboxName.methodName (parameters)
- formName.elements[i].propertyName
- formName.elements[i].methodName (parameters)

де checkboxName – значення атрибуту name об'єкту checkbox, а formName – ім'я об'єкту form або форми, якій належить даний контрольний перемикач. Іншими словами, до форми можна звертатися як до елементу масиву forms, наприклад forms[0] - для звернення до першої форми документа, або на ім'я об'єкту form, якщо воно визначене в атрибуті name HTML-тегу <form>.

До будь-якого елемента форми можна звернутися так само, як до елементу масиву elements, який є властивістю об'єкту form. У цьому випадкові для звертання до певного контрольного перемикача слід використовувати його порядковий номер (i) у масиві елементів форми.

Властивості об'єкта checkbox

Властивість	Опис	IE	F	O	W3C
accessKey	Встановлює або повертає кнопку доступу на клавіатурі до об'єкта	4	1	9	Yes
alt	Встановлює або повертає текст, що виводиться у браузерах, які не підтримують об'єкт	5	1	9	Yes
checked	Встановлює, чи вибрати checkbox, або повертає відповідне значення	4	1	9	Yes
defaultChecked	Повертає атрибут checked по замовчуванню	4	1	9	Yes
disabled	Встановлює або повертає значення: чи елемент	4	1	9	Yes

Властивість	Опис	IE	F	O	W3C
	деактивований				
form	Повертає посилання на форму, якій належить елемент	4	1	9	Yes
id	Встановлює або повертає id елемента	4	1	9	Yes
name	Встановлює або повертає name елемента	4	1	9	Yes
tabIndex	Встановлює або повертає tab-порядок елементів	4	1	9	Yes
type	Повертає тип елемента форми, яким є checkbox	4	1	9	Yes
value	Встановлює значення атрибута value елемента	4	1	9	Yes

Стандартні властивості

Властивість	Опис	IE	F	O	W3C
className	Встановлює або повертає значення атрибуту class	5	1	9	Yes
dir	Встановлює або повертає значення напряму тексту	5	1	9	Yes
lang	Встановлює значення або повертає код мови елемента	5	1	9	Yes
title	Встановлює або повертає title елемента	5	1	9	Yes

Методи об'єкта checkbox

Метод	Опис	IE	F	O	W3C
blur()	Видаляє фокус з checkbox	4	1	9	Yes
click()	Симулює клік кнопки миші на елементові checkbox	4	1	9	Yes
focus()	Встановлює фокус на checkbox	4	1	9	Yes

4.5 Масив elements

Масив elements містить усі елементи HTML-форми: контрольні перемикачі (checkbox), селекторні кнопки (radio-button), текстові об'єкти (text) та інші у тій послідовності, у якій вони визначені у формі. Цей масив використовують для доступу до елементів форми у JS-програмі за їхнім порядковому номером, не використовуючи властивості name цих елементів. Масив elements, у свою чергу, є властивістю об'єкту forms, тому при звертанні до нього слід вказувати ім'я форми, до елементу якої потрібно звернутися:

```
formName.elements[i],
formName.elements[i].length .
```

Тут formName може бути або ім'ям об'єкту form, визначеним за допомогою атрибуту name у тегові <form>, або елементом масиву forms, наприклад forms[i], де i – змінна, яка індексує елементи масиву. Значенням властивості length є кількість елементів, що містяться у формі. Масив elemments включає дані тільки для читання.

4.6 Об'єкт form і масив forms

Форма – це область гіпертекстового документу, яка створюється за допомогою контейнера <form> . . . </form> і містить елементи, що дозволяють користувачеві вводити інформацію. HTML-теги, що визначають поля введення (text field), області тексту (textarea), контрольні перемикачі (checkbox), селекторні кнопки (radio button) і списки (selection list), розташовуються тільки в контейнері <form> . . . </form>.

Даним елементам у мові JavaScript відповідають окремі об'єкти. Програми на мові JS можуть обробляти форми безпосередньо, отримуючи значення, що міститься у необхідних елементах (наприклад для перевірки введення обов'язкових даних). Крім того, дані з форми звичайно передаються для обробки на віддалений Web-сервер. Синтаксис:

```
<form name="formName"
target="windowname"
action="serverURL"
method="get" | "post"
enctype="encodingType"
[onSubmit="handlerText"]>
</form>
```

Тут атрибут `name` – рядок, що визначає ім'я форми. Атрибут `target` задає ім'я вікна, в якому повинні оброблятися події, пов'язані зі зміною елементів форми. Для цього потрібна наявність вікна або фрейма із заданим ім'ям. Як значення даного атрибуту можуть використовуватися і зарезервовані імена `_blank`, `_parent`, `_self` і `_top`.

Атрибут `action` задає адресу URL сервера, який одержуватиме дані з форми і запускати відповідний CGI-скрипт. Також можна послати дані з форми електронною поштою, вказавши при цьому значення цього атрибуту адреса URL типу `mailto: . .`

Форми, що передаються на сервер, вимагають вказання методу передачі (`submission`), який вказується за допомогою атрибуту `method`. Метод `GET` приєднує дані форми до рядка адреси URL, заданої в атрибуті `action`. При використанні методу `POST` інформація з форми посилається як окремий потік даних. У останньому випадку CGI-скрипт на сервері прочитує ці дані із стандартного вхідного потоку (standard input stream). Крім того, на сервері встановлюється змінна середовища з ім'ям QUERY_STRING, що забезпечує ще один спосіб отримання цих даних.

Атрибут `enctype` задає тип кодування MIME (Multimedia Internet Mail Extensions) для посиланих даних. Типом MIME за умовчанням є тип application/x-www-form-urlencoded.

До властивостей і методів форми у JavaScript-програмі можна звернутися одним із способів:

- `formName.propertyName;`
- `formName.methodName (parameters);`
- `forms[i].propertyName;`
- `forms[i].methodName (parameters).`

Тут `formName` відповідає атрибуту `name` об'єкту `form`, і є цілочисленою змінною, що використовується для звертання до окремого елементу масиву `forms`, який відповідає визначеному тегу `<form>` поточного документа.

Колекції об'єкту `Form`

Колекція	Опис	**IE**	**F**	**O**	**W3C**
`elements[]`	Повертає масив, який містить елементи форми	5	1	9	Yes

Властивості об’єкту `Form`

Властивість	**Опис**	**IE**	**F**	**O**	**W3C**
`acceptCharset`	Встановлює або повертає список можливих наборів символів даних форми	No	No	No	Yes
`action`	Встановлює або повертає атрибут action форми	5	1	9	Yes
`enctype`	Встановлює або повертає MIME-тип, що використовується для кодування елементів форми	6	1	9	Yes
`id`	Встановлює або повертає id форми	5	1	9	Yes
`length`	Повертає кількість елементів форми	5	1	9	Yes
`method`	Встановлює або повертає HTTP метод надсилання даних на сервер	5	1	9	Yes
`name`	Встановлює або повертає ім’я форми	5	1	9	Yes
`target`	Встановлює або повертає, де відкрити action-URL форми	5	1	9	Yes

Стандартні властивості

Властивості	**Опис**	**IE**	**F**	**O**	**W3C**
`className`	Встановлює або повертаєатрибут class елемента	5	1	9	Yes
`dir`	Встановлює або повертає напрям тексту елемента	5	1	9	Yes
`lang`	Встановлює або повертає код мови елемента	5	1	9	Yes
`title`	Встановлює або повертає title елемента	5	1	9	Yes

Методи об’єкту `Form`

Методи	**Опис**	**IE**	**F**	**O**	**W3C**
`reset()`	Скидає значення елементів форми	5	1	9	Yes
`submit()`	Надсилає форму	5	1	9	Yes

4.7 Масив `frames`

До окремих фреймів можна звертатися за допомогою масиву `frames` і властивості `parent`. Наприклад, якщо є два фрейми, визначені в HTML-тегах:

```
<frameset rows="50%,50%">
<frame name="top" src="file1.htm">
<frame name="bot" src="file2.htm">
</frameset>
```

то для звертання до першого фрейму можна використовувати вираз `parent.frames[0]`, і відповідно, до другого – `parent.frames[1]`. Таким чином, для звертання до окремих фреймів і до властивості `length` масиву `frames` використовуються вирази вигляду:

- `frameRef.frames[i];`
- `frameRef.frames.length;`
- `windowRef.frames[i];`
- `windowRef.frames.length`.

Для визначення кількості фреймів у документі використовують властивість `length`. Усі дані масиву `frames` призначені тільки для читання.

4.8 Прихований об’єкт

Це поле, яке може передаватися з форми наприклад на сервер, знаходитися у тегах `<form> . . . </form>`, і при цьому не відображатися на екрані. Ці поля дозволяють зберігати певні значення у структурах, відмінних від змінних мови JS, хоча дані значення існують до тих пір, поки завантажений поточний документ.

HTML-тег має синтаксис:

```
<input type="hidden"
 [name="hiddenName"]
 [value="textValue"]>
```

Атрибут `name` задає ім’я поля і є необов’язковим. Значення текстового поля вказують за допомогою атрибуту `value`, який дозволяє задавати і значення поля по замовчуванню. До властивостей прихованих об’єктів можна звертатися за допомогою одного з наступних виразів:

- `fieldName.propertyName`
- `formName.elements[i].propertyName`

де `fieldName` – ім’я прихованого поля, задане в атрибуті name тега `<input>`, а `formName` – ім’я форми, в якій визначено приховане поле.

Властивості. Прихований об’єкт має властивості:

– `name` – відповідає атрибутові `name` тегу `<input>`;
– `value` – відповідає атрибутові `value` тегу `<input>`;
– `type` – відповідає атрибутові `type` і містить значення `"hidden"`.

Приховані об'єкти не мають методів і обробників подій.

Приклад

У наступній формі визначено приховане поле hfield шириною 20 символів, по замовчуванню має значення "page 1":

```
<form name="hiddenField">
<input name="hfield" type="hidden" size=20 value="page 1">
</form>
```

Значення цього поля можна змінити за допомогою оператора наступного вигляду:

```
document.hiddenField.hfield.value = "page 2".
```

4.9 Об'єкт image і масив images

У браузері малюнки розглядають як об'єкти image, а всі малюнки, що містяться в поточному документі, поміщають у масив images, який можна використовувати для звернення до будь-якого малюнка, який визначається тегом <img>. Зокрема, можна динамічно оновлювати зображення, змінюючи їхні властивості src. Синтаксис тегу <img> наступний:

```
<img src="fileOrUrl"
 [alt="altText"]
 [lowsrc="fileOrUrl"]
 [width=w]
 [heigth=h]
 [border=b]
 [vspace=v]
 [hspace=h]>
```

Атрибут src містить ім'я або адресу URL файлу, який потрібно вивести в документі. Малюнок повинен зберігатися у форматі GIF, JPEG, або PNG. За допомогою атрибуту alt задається альтернативний текст, що з'являється на екрані: у момент завантаження тексту, якщо користувач заблокував виведення зображень і що пояснює напис під курсором миші. Атрибут lowsrc, NN його підтримує, IE - чесно сказати не знаю, не пробував, та і без потреби - вважаю застарілим і не має сенсу його використовувати. Він дозволяє заздалегідь виводити на екран зображення з низьким дозволом. При цьому малюнок завантажується в два етапи. Атрибути width (ширина) і height (висота) дозволяють задати розміри малюнка в пікселах, атрибут border - ширину рамки в пікселах, а атрибути vspace і hspace - розміри вертикального і горизонтального зазорів між межами зображення і іншими елементами документа.

Для звернення до властивостей об'єкту image використовується наступний синтаксис:

```
document.images[i].propertyName
```

де i - індекс елементу масиву, який відповідає потрібному малюнку. Першим малюнком в документі буде document.images[0]. Масив images є властивістю об'єкта document, тому при звертанні до малюнка необхідний префікс document до імені масиву. Тег <img> не має атрибуту name, тому вираз виду "document.imgName" приведе до помилки.

Властивості об'єкту Image

Властивість	Опис	IE	F	O	W3C
align	Встановлює або повертає значення, що відповідає розташуванню об'єкта щодо оточуючого його тексту	5	1	9	Yes
alt	Встановлює або повертає текст, що виводиться у випадкові неможливості відображення елемента	5	1	9	Yes
border	Встановлює або повертає значення рамки навколо малюнка	4	1	9	Yes
complete	Встановлює, чи браузер завершив завантаження зображення	4	1	9	No
height	Встановлює або повертає висоту зображення	4	1	9	Yes
hspace	Встановлює або повертає білий простір зліва та справа від зображення	4	1	9	Yes

Властивість	Опис	IE	F	O	W3C
id	Встановлює або повертає id зображення	4	1	9	Yes
isMap	Встановлює, чи зображення є частиною image map на боці сервера	5	1	9	Yes
longDesc	Встановлює або повертає шлях URL до документа, що містить опис зображення	6	1	9	Yes
Lowsrc	Встановлює або повертає шлях URL до версії зображення із низькою роздільною здатністю	4	1	9	No
name	Встановлює або повертає name зображення	4	1	9	Yes
src	Встановлює або повертає URL зображення	4	1	9	Yes
useMap	Встановлює або повертає значення атрибуту usemap для image map на боці сервера	5	1	9	Yes
vspace	Встановлює або повертає білий простір згори та знизу від зображення	4	1	9	Yes
width	Встановлює або повертає ширину зображення	4	1	9	Yes

Стандартні властивості

Властивість	Опис	IE	F	O	W3C
className	Встановлює або повертає class-атрибут елемента	5	1	9	Yes
title	Встановлює або повертає title елемента	5	1	9	Yes

4.10 Об’єкт link і масив links

Об’єкт link (гіперпосилання) відображається як ділянка тексту або графічного об’єкту, клацання миші на якому дозволяє перейти до іншого Web-ресурсу. Тег мови HTML має наступний вигляд:

```
<a href=locationOrURL
[name="anchorName"]
[target="windowOrFrameName"]
[onClick="handlerText"]
[onMouseOver="handlerText"]>
linkText
</a>
```

Атрибут href визначає ім’я файлу або адресу URL для об’єкту, який завантажується при активізації гіперпосилання. Атрибут name задає ім’я гіперпосилання, перетворюючи його на об’єкт anchor (мітку). За допомогою атрибуту target у певний фрейм поточного документа можна завантажити документ, URL якого вказаний у значенні атрибуту href. Атрибут linkText – це текст, що відображається у HTML-документі як гіперпосилання, яке активізується клацанням миші. Для звертання до властивості об’єкту link використовуються вирази типу:

```
document.links[i].propertyName ,
```

де i - індекс даного гіперпосилання у масиві links поточного документу.

Властивості об’єкту `Link`

Властивість	Опис	**IE**	**F**	**O**	**W3C**
`charset`	Встановлює або повертає кодування символів для цільового URL	4	1	9	Yes
`disabled`	Встановлює або повертає, чи цільовий URL повинен бути деактивований	4	1	9	Yes
`href`	Встановлює або повертає URL зв’язаного ресурсу	4	1	9	Yes
`hreflang`	Встановлює або повертає базову мову цільового URL	4	1	9	Yes
`id`	Встановлює або повертає id елементу `<link>`	4	1	9	Yes
`media`	Встановлює або повертає, який пристрій відтворюватиме документ	6	1	9	Yes
`name`	Встановлює або повертає name-параметр елемента <link>	4	No	No	Yes
`rel`	Встановлює або повертає взаємозв’язок між поточним документом та цільовим URL	4	1	9	Yes
`rev`	Встановлює або повертає взаємозв’язок між цільовим URL та поточним документом	4	1	9	Yes
`type`	Повертає значення MIME-типу цільового URL	4	1	9	Yes

Стандартні властивості

Властивість	Опис	**IE**	**F**	**O**	**W3C**
`dir`	Встановлює або повертає напрям тексту	5	1	9	Yes
`lang`	Встановлює або повертає код мови елемента	5	1	9	Yes

Масив `links`

У програмі на мові JavaScript до гіперпосилань можна звертатися, як до елементів масиву `links`. Наприклад, якщо в документі визначені два теги `< a href>`, то в JS-програмі до цих гіперпосилань можна звертатися за допомогою виразів `document.links[0]` і `document.links[1]`. Синтаксис виразів для звертань до масиву `links` наступний:

- `document.links[i];`
- `document.links.length`,

де змінна `i` - індекс гіперпосилання. Значенням властивості `length` є кількість гіперпосилань у поточному документі. Об’єкти `link` є об’єктами тільки для читання, тому динамічно змінювати гіперпосилання у документі не можна.

5 Система подій мови JavaScript

Використання мови JavaScript при обробці подій значно розширило можливості мови HTML. Можливості керування елементами форм забезпечуються головним чином за рахунок функцій обробки подій, які можуть бути задані для всіх елементів форми. Події діляться на декілька категорій:

– події, пов’язані з документами (події документа);
– завантаження і вивантаження документів;
– події, пов’язані з гіперпосиланням (події гіперпосилання;
– активізація гіперпосилання;
– події, пов’зані з формою (події форми);
– клік миші по кнопках;
– отримання і втрата фокусу введення і зміна вмісту полів введення, областей тексту та списків;
– виділення тексту в полях введення і областях тексту;
– події, пов’язані з мишею;
– наведення курсору миші на гіперпосилання і активізація гіперпосилання.

Події, пов’язані з документами, виникають при завантаженні і вивантаженні документа, тоді як події гіперпосилань виникають при їх активізації або при наведення на них курсору миші.

Для забезпечення перехоплення події, необхідно написати функцію-обробник події. Як обробники подій можуть бути задані цілі функції мови JavaScript або тільки групи з одного або декількох JS-операторів. Нижче наведено імена подій та умови їх виникнення:

Ім’я події	Атрибут HTML	Умова виникнення події
`Blur`	`onBlur`	Втрата фокусу введення елементом форми
`Change`	`onChange`	Зміна вмісту поля введення йди області тексту, або вибір нового елементу списку
`Click`	`onClick`	Клацання миші на елементі форми або гіперзв'язка
`Focus`	`onFocus`	Отримання фокусу введення елементом форми
`Load`	`onLoad`	Завершення завантаження документа
`MouseOver`	`onMouseOver`	Приміщення покажчика миші на гіперзв'язок
`MouseOut`	`onMouseOut`	Приміщення покажчика миші не на гіперзв'язок
`Select`	`onSelect`	Виділення тексту в полі введення або області тексту
`Submit`	`onSubmit`	Передача даних форми
`Unload`	`onUnload`	Вивантаження поточного документа і почало завантаження нового

5.1 Втрата фокусу – подія `onBlur`

Атрибут обробника події onBlur працює відповідає наступними тегам і html:

```
<input.type=". . ." onBlur="expr | функція()">
 <textarea onBlur="expr | функція()"> . . . </textarea>
```

```
<select onBlur="expr | функція()"> . . . <option> . . .
</select>
```

За допомогою атрибуту `onBlur` задається вираз мови JavaScript, який виконується, коли відповідний елемент HTML-форми втрачає фокус введення. Втрата фокусу введення відбувається або при клацанні миші на іншому елементі форми або іншої форми, або при переході до іншого елементу форми за допомогою клавіші [tab]. Атрибут `onBlur` застосовують для перевірки даних, введених у відповідне поле.

5.2 Подія `onChange` – зміна вмісту поля і вибраних елементів списку

Атрибут обробника події `onChange` можна використовувати в наступних HTML-ТЕГАХ:

```
<select onChange="expr | функція()"> . . . <option> . . .
</select>
<textarea onChange="expr | функція()"> . . . </textarea>
<input type=text onChange="expr | функція()">
```

Атрибут `onChange` задає вираз, який повинен виконуватися при втраті фокусу введення елементом HTML-форми і при зміні вмісту цього елементу. Даний атрибут подібний до атрибуту `onBlur`, проте для того, щоб виникла подія `Change`, вміст поля повинен бути змінений, і поле повиннj втратити фокус введення.

Розглянемо невеликий приклад:

```
<script type ="text/javascript">
<!--
function selChange(seln){
selNum = seln.beer.selectedIndex;
Isel = seln.beer.options[selNum].text;
alert("ВИБРАНО: "+Isel);
}
//-->
</script>

<form>
Виберіть марку автомобіля:
<select name="car" onChange="selChange(this.form) ">
<option>ЗАЗ
<option>ГАЗ
<option>УАЗ
<option>КРАЗ
</select>
</form>
```

У цьому прикладі є єдиний об’єкт Select з іменем car, що містить чотири елементи, визначені в тегах <option>. Кожного разу при виборі нового елементу викликається функція JavaScript з іменем `selChange()`. Для звертання до значень тегів `<option>` у програмі

використовується масив властивостей options. Подібний приклад розбирався раніше, тільки зараз використовується подія onChange.

5.3 Подія onClick – активізація гіперпосилань

Атрибут onClick може використовуватися в наступних тегах HTML:

```
<a herf=URL onClick="expr | function()">. . .</a>
<input.type="checkbox" onClick="expr | function()">
<input.type="radio" onClick="expr | function()">
<input.type="reset" onClick="expr | function()">
<input.type="submit" onClick="expr | function()">
<input.type="button" onClick="expr | function()">
```

Оператори мови JavaScript, задані в атрибуті onClick, виконуються при клацанні миші на таких об'єктах як гіперпосилання, кнопка перезавантаження форми або контрольний перемикач. Для контрольних перемикачів і селекторних кнопок подія Click виникає не тільки при виборі елементу, але і при розблокуванні.

Розглянемо приклад використання атрибуту onClick для кнопок, визначених тегами <input type="button"> у контейнері <form> . . . </form>:

```
<script language="JavaScript">
<!--
function but1() {
alert("Ви натиснули першу кнопку");
}
function but2() {
alert("Ви натиснули другу кнопку");
}
//-->
</script>
<form>
<input type="button" value="Перша кнопка" onClick="but1()">
<input type="button" value="Друга кнопка" onClick="but2()">
</form>
```

Коли користувач клацає мишею по кнопці, викликається або функція but1(), або but2(). При цьому за допомогою методу alert() на екран виводиться відповідне повідомлення.

Розберемо використання події onClick наприклад для контрольних перемикачів. Обробка цієї події тут виконується як при включенні, так і при виключенні контрольних перемикачів. Для перевірки стану перемикачів слід використовувати властивість checked, яка містить значення true, якщо перемикач включений. Розглянемо приклад:

```
<script language="JavaScript">
<!--
function chk1(f) {
if (f.checked)
```

```
      alert("Перший перемикач включений");
      else
      alert("Перший перемикач відключений");
      }
      function chk2(f) {
      if (f.checked)
      alert("Другий перемикач включений");
      else
      alert("Другий перемикач відключений");
      }
      //-->
      </script>
      <form name="chkform">
      <input type="checkbox" checked name="c1"
onClick="chk1(this.form.c1)"> Перемикач 1 <BR>
      <input type="checkbox" name="c2" onClick="chk2(this.form.c2)">
Перемикач 2
      </form>
```

після передачі значення `this.form.c1` або `this.form.c2` у відповідні функції перевіряється властивість `checked` контрольного перемикача, переданого функції, і залежно від значення властивості checked спрацьовує метод `alert()`.

Звичайно подібний підхід застосовується для складного введення якихось даних, і залежно від встановленого "прапорця" пропонується заповнювати відповідні поля форм або перемістити фокус в одне з полів, оскільки дані відстежуються ще при введенні поточної форми.

5.4 Отримання фокусу введення – подія `onFocus`

Атрибут обробника події onFocus працює з наступними тегами HTML:

```
<input.type="text" onFocus="expr | function()">
<textarea onFocus="expr | function()"> . . . </textarea>
<select onFocus="expr | function()"> . . . <option> . . .
</select>
```

Назва говорить за себе, атрибут onFocus дозволяє обробляти події, пов’язані з отриманням фокусу введення. На відміну від onBlur, тут обробляється подія при отриманні фокусу введення.

Атрибут обробника події onLoad працює з наступними тегами HTML:

```
<body onLoad="expr | function()"> . . . </body>
<frameset> . . . <frame onLoad="expr | function()"> . . .
</frameset>
```

5.5 Подія onLoad – завантаження документа

Атрибут onLoad, поміщений в тег <body>, активізує задані операторів мови JavaScript, коли завантаження поточного документа у браузер завершене. Подія відбувається після завершення завантаження тексту HTML у поточне вікно або фрейм.

Для чого це потрібно? Іноді користувач не дочекавшись завантаження всього документа переходить по посиланню в наступний. Може бути така ситуація, що заздалегідь потрібно ввести якісь дані в поточному документі, поля яких ще не завантажені. Подія onLoad не може використовуватися для зміни поточного документа, але за допомогою відповідних функцій можна наприклад залишити дані в cooke. Також для гарантованого відробітку яких-небудь функцій на мові JavaScript в поточному документі.

5.6 Подія unLoad – вивантаження документа

Атрибут обробника події unLoad працює з наступними тегамі HTML:

```
<body unLoad="expr | function()"> . . . </body>
<frameset> . . . <frame unLoad="expr | function()"> . . .
</frameset>
```

Відповідна подія виникає при вивантаженні поточного документа, тобто викликається функція-обробник події перед вивантаженням документа з поточного вікна або фрейма.

Для чого це може стати в нагоді? Уявимо, що користувач в останню мить відмовився надіслати форму і кликнув на яку або посилання або банер, а нам потрібно знати якусь, наприклад для статистики, інформацію введену користувачем, – тут можна перевірити, чи форма не надсилалася і чи не порожні поля, надіслати без підтвердження з відповідною позначкою. Оскільки подія виникає перед вивантаженням, то цей обробник можна використовувати, наприклад, для контролю необхідного введення реєстраційної інформації і ін.

5.7 Події onMouseOver і onMouseOut – переміщення миші

OnMouseOver дозволяє активізувати JavaScript-оператори, коли курсор миші знаходиться на активному гіперпосиланні, а атрибут onMouseOut – коли курсор відведений від гіперпосилання.

```
<a href=". . . " onMouseOver="expr | function()"> . . . </a>
<a href=". . . " onMouseOut="expr | function()"> . . . </a>
```

Атрибут onMouseOver викликає оператори JavaScript, коли курсор миші наведений на гіперпосилання, в якому заданий цей атрибут. При обробці подій MouseOver можна змінювати повідомлення в рядках стану і текстові поля. Крім того часто використовуються для динамічної підміни малюнка в навігаційній панелі.

Атрибут onMouseOut надає можливість активізувати оператори мови JavaScript за допомогою курсора миші за межі області гіперпосилання. Обробку події onMouseOut слід виконувати, коли необхідно відмінити раніше задані дії. Наприклад, якщо при обробці події MouseOver відбулася підміна малюнка з масиву images, то при обробці події onMouseOut можуть бути востановлени початкові дані.

Короткий опис атрибуту.

```
<script language ="JavaScript">
<!--
function clearf() {
document.forms[0].atr_text.value = "Короткий опис атрибуту";
}
var atrText1=
"Атрибут OnMouseOver-приміщення миші на гіперпосилання\r\n\n"+
"Далі може знаходитися докладний опис цього\r\n"+
"атрибуту обробника подій";
var atrText2=
"Атрибут OnMouseOut-обробка події при переміщенні\r\n"+
"миші за межі гиперпосилання\r\n\n"+
"Далі може знаходитися докладний опис цього\r\n"+
"атрибуту обробника подій";
//-->
</script>
```

Для отримання докладнішої інформації помістите покажчик миші на гіперпосилання.

```
<a href="#"
onMouseOver="document.forms[0].atr_text.value=atrText1"
onMouseOut=clearf()>OnMouseOver</a>
<a href="#"
onMouseOver="document.forms[0].atr_text.value=atrText2"onMouseOut=cle
arf()>OnMouseOut</a>
<form>
<textarea rows=5 cols=60 name="atr_text" wrap="soft" >
Тут приведено короткий опис атрибуту
</textarea>
</form>
```

На основі цієї програми можна створити інші документи і включити їх у власну обробку різних подій.

5.8 Події `onSelect` – виділення тексту

Атрибут onSelect може бути використаний з наступними тегами:

```
<input.type="text" onSelect="expr | function()">
<textarea onSelect="expr | function()"> . . . </textarea>
```

Цей атрибут запускає обробник події, коли користувач виділив фрагмент тексту в полі введення або області тексту.