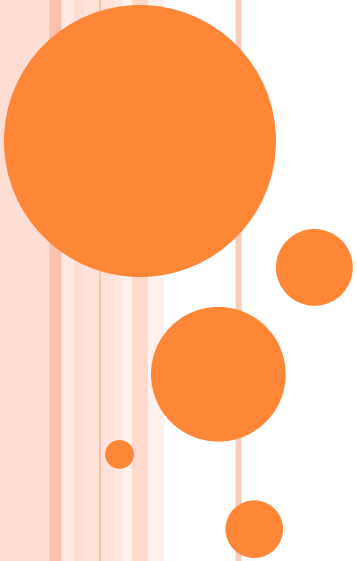


Лекція 31

Користувальницькі об'єкти

Поняття призначеного для користувача об'єкта

- function Rectangle (a, b, c, d)
- {
- this.x0 = a;
- this.y0 = b;
- this.x1 = c;
- this.y1 = d;
- this.area = new Function (
- "Return Math.abs ((this.x1-this.x0) * (this.y1-this.y0))");
- }
- r = **new Rectangle (0,0,30,50);**
- Функція Rectangle () - це конструктор об'єкта класу Rectangle, визначеного користувачем. Конструктор дозволяє створити екземпляр (об'єкт) даного класу. Адже функція - це не більше ніж опис деяких дій. Для того щоб ці дії були виконані, необхідно передати функції управління. У нашому прикладі це робиться за допомогою оператора new Rectangle.

Прототип

- Зазвичай ми маємо справу з вбудованими об'єктами JavaScript, такими як Data, Array і String.
- У цьому сенсі цікаво одна властивість об'єктів, яке носить назву prototype.
- ***Прототип*** - це інша назва конструктора об'єкта конкретного класу.
- Наприклад, якщо ми хочемо додати метод до об'єкту класу String, то ми можемо це зробити наступним чином:
- **String.prototype.out = new Function ("a", "a.write (this)");**
- var s = "Привіт!";
- s.out (document);
- // Буде виведено: Привіт!
- Є один істотний нюанс: новими методами і властивостями будуть володіти тільки ті об'єкти, які породжуються після зміни прототипу об'єкта.
- Всі вбудовані об'єкти створюються до того, як JavaScript-програма отримає управління, що істотно обмежує застосування властивості prototype.

Прототип

- Проте покажемо, як можна додати метод до вбудованого в JavaScript класу. *Завдання полягає в тому, щоб розібрати URL картинки таким же чином, як і URL об'єкта класу Link, тобто забезпечити об'єкт класу Image додатковими методами protocol (), host () і т.п .*:
- Додавання методів до класу Image
- function pr ()
- {
- a = this.src.split (':');
- return a [0] + ':';
- }
- function ho ()
- {
- a = this.src.split (':');
- path = a [1] .split ('/');
- return path [2];
- }
- function pa ()
- {
- path = this.src.split ('/');
- path [0] = '';
- path [2] = '';
- return path.join ('/'). split ('///'). join ('/');
- }
-

Прототип

- Image.prototype.protocol = pr;
- Image.prototype.host = ho;
- Image.prototype.pathname = pa;
-
- document.write ("<IMG NAME = i1 SRC = 'image1.gif'>
");
- document.write (document.i1.src + "
");
- document.write (document.i1.protocol () + "
");
- document.write (document.i1.host () + "
");
- document.write (document.i1.pathname () + "
");

Методи об'єкта Object

- ***Object*** - це клас, елементами якого *є будь-які об'єкти JavaScript*. У всіх об'єктів цього класу є загальні методи. Таких методів ми розглянемо три: toString (), valueOf () і assign ().
- Метод toString () здійснює перетворення об'єкта в рядок символів (строковий літерал). Він використовується в JavaScript-програмах повсюдно, але в ***основному неявно.***
- Наприклад, при виведенні числа або строкових об'єктів. document.write (pr.toString ());
- Результат виконання:
- function pr ()
- {
- a = this.src.split (':');
- return a [0] + ':';
- }
- Однак, якщо роздрукувати таким же чином об'єкт класу Image з того ж прикладу:
- document.write (document.i1.toString ());
- то отримаємо вже наступне: [object Image] ().

Методи об'єкта Object

- Метод valueOf () - дозволяє отримати значення об'єкта. У більшості випадків він працює подібно методу toString (), особливо якщо потрібно виводити значення на сторінку.
- Наприклад, оператор
- document.write (pr.valueOf ())
- document.write (pr.toString ()) – з попереднього слайду

- Метод assign () - дозволяє не прочитати, а перепризначити якусь властивість і метод об'єкта. Цей метод працює не у всіх браузерах і не з усіма об'єктами.
- У загальному випадку має вигляд:
- об‘єкт.властивість = значення рівносильне оператору об'ект.свойство.assign (значення).
- Наприклад, такі оператори рівносильні - вони перенаправляють користувача на нову сторінку:
- window.location = "http://intuit.ru/";
- window.location.assign ("http://intuit.ru/");

Об'єкт WINDOW

- Клас об'єктів Window - *це найстарший клас в ієрархії об'єктів JavaScript.*
- Об'єкт window, що відноситься до поточного вікна (тобто в якому виконується скрипт), є об'єктом класу Window.

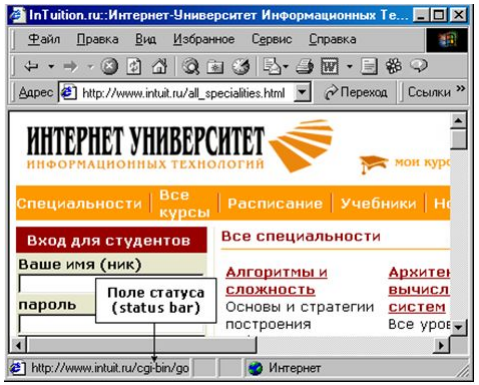
- Об'єкт window створюється ***тільки*** в момент відкриття вікна.
- Всі інші об'єкти, які породжуються при завантаженні сторінки, є властивості об'єкта window.
-
- Більш того, всі глобальні змінні, визначені в даному вікні, теж є властивостями об'єкта window.

Властивості, методи та події об'єкта window

Властивості	Методи	Події
status	open()	Load
defaultstatus	close()	Unload
location	focus()	Focus
history	blur()	Blur
navigator	alert()	Resize
document	confirm()	Error
frames[]	prompt()	
opener	setTimeout()	
parent	setInterval()	
self	clearTimeout()	
top	clearInterval()	

Властивості об'єкта window

- ***Поле статусу і властивість window.status***
- Поле статусу - це перше, що почали використовувати автори HTML-сторінок з арсеналу JavaScript.



- Програма на JavaScript має можливість працювати з цим полем як із змінним властивістю вікна. При цьому фактично з ним пов'язані два різних властивості:
- • window.status - значення поля статусу;
- • window.defaultStatus - значення поля статусу за замовчуванням.
- Значення властивості status можна змінити - і воно тут же буде відображено в полі статусу.
- Властивість defaultStatus теж можна міняти - і відразу по його зміні воно відображається в полі статусу.

Програмуємо STATUS

- Властивість status пов'язане з відображенням повідомлень про події, відмінних від простого завантаження сторінки.
- Приклади:
- <A
- onMouseOver = "window.status = 'Миша над посиланням'; return true;
- onMouseOut = "window.status = 'Миша повели з посилання';
- href = "http://site.com/"> Наведіть мишу на посилання і стежте за полем статусу
- </A>
- Зверніть увагу на оператор return true в кінці обробника подій onMouseOver. Він необхідний для того, щоб скасувати дію за замовчуванням (в даному випадку - висновок URL в поле статусу), яке, за відсутності цього оператора, браузер виконав би відразу після виведення нами своєю рядки в поле статусу, і користувач не встиг би побачити нашу рядок .

Програмуємо defaultstatus

- Властивість defaultStatus визначає текст, який відображається в полі статусу, коли ніяких подій не відбувається.
- <body onload = "window.defaultstatus = 'значення за замовчуванням';">
- <a onmouseover = "window.status = 'миша над посиланням'; return true;"
- onmouseout = "window.status = 'миша повели з посилання'; alert ('чекаємо');"
- href = "http://site.com/"> наведіть мишу на посилання і стежте за полем статусу
- </ body>

- Відразу після завантаження документа в поле статусу буде "Значення за замовчуванням". При наведенні покажчика миші на посилання в поле статусу з'явиться напис "Миша над посиланням", при цьому URL посилання (http://site.com/) в поле статусу не з'явиться, тому що ми придушили його висновок оператором return true.

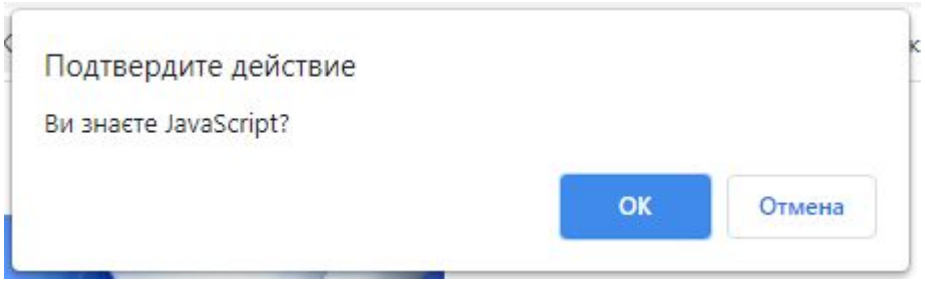
Методи об'єкта window

- ***alert ()*** - дозволяє видати вікно попередження, що має єдину кнопку "OK":
- <button onClick="javascript:window.alert('Увага')"> again! </button>

- Потрібно лише мати на увазі, що повідомлення виводяться системним шрифтом, отже, для отримання попереджень російською мовою потрібна локалізована версія ОС.

Методи об'єкта window

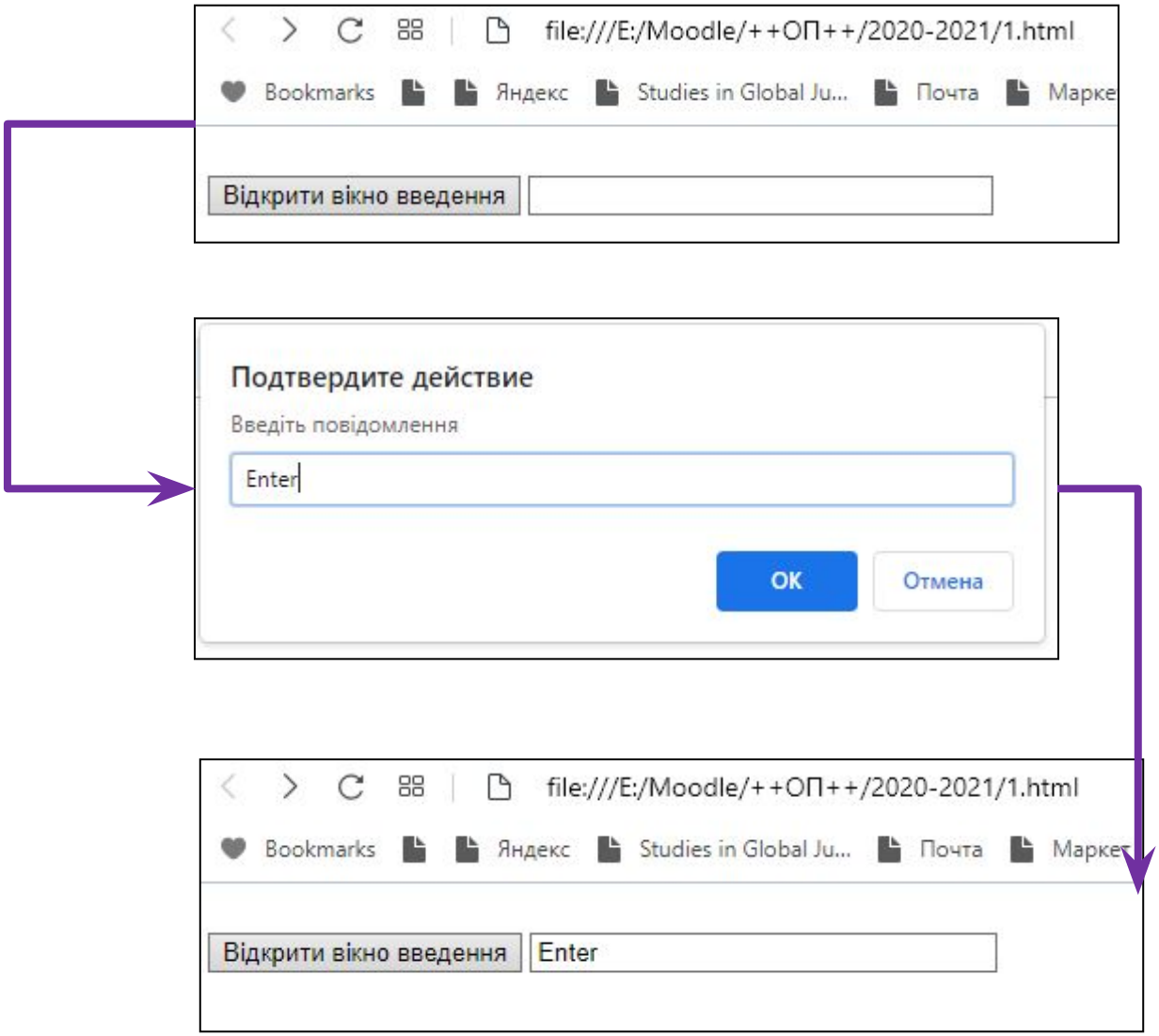
- ***confirm ()*** – дозволяє задати користувачеві питання, на який той може відповісти або позитивно (натиснувши кнопку "OK"), або негативно (натиснувши кнопку "Скасувати" або "Cancel", або просто закривши вікно запиту). Відповідно до діями користувача метод confirm () повертає значення true або false. приклад:
- <form name = f>
- <input type = button name = b value = "натисніть цю кнопку"
- onClick = "if (window.confirm ('Ви знаєте JavaScript?'))
- document.f.b.value = 'Так. Запитати ще? ';
- else document.f.b.value = 'Немає. Запитати ще? '; ">
- </ form>



Методи об'єкта window

- ***prompt ()*** – дозволяє прийняти від користувача Рядок тексту.
- Синтаксис його такий:
- *prompt ("Рядок питання", "Рядок відповіді за замовчуванням")*
- Коли користувач введе свій відповідь (або залишить незмінним відповідь за замовчуванням) і натисне кнопку OK, метод prompt () поверне отриману рядок в якості значення, яке можна далі привласнити будь-якої змінної і потім розбирати її в JavaScript-програмі.
- <form name = f>
- <input type = button value = "Відкрити вікно введення"
- onClick = "document.f.e.value =
- window.prompt ('Введіть повідомлення', 'Сюди'); ">
- <input size = 30 name = e>
- </ form>
-

Методи об'єкта window



Методи об'єкта window

- ***window.open()*** – призначений для створення нових вікон.
- У загальному випадку його синтаксис виглядає наступним чином:
- *myWin = window.open ("URL", "ім'я_окна", "параметр = значення, параметр = значення, ...", замінити);*
- Перший аргумент задає адресу сторінки, що завантажується в нове вікно (можна залишити порожній рядок, тоді вікно залишиться порожнім).
- Другий аргумент задає ім'я вікна, яке можна буде використовувати в атрибуті TARGET контейнерів <A> і <FORM>. Як значення допустимі також зарезервовані імена _blank, _parent, _self, _top, зміст яких такий же, як у аналогічних значень атрибута TARGET. Якщо ім'я_окна збігається з ім'ям вже існуючого вікна (або фрейма), то нове вікно не створюється, а всі наступні маніпуляції зі змінною myWin будуть застосовуватися до цього вікна (або кадру).

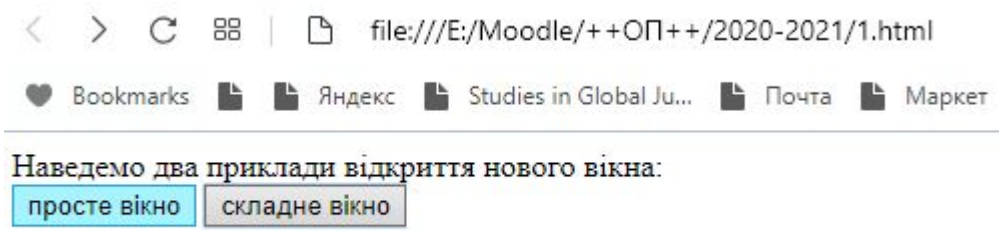
Параметры метода window.open()

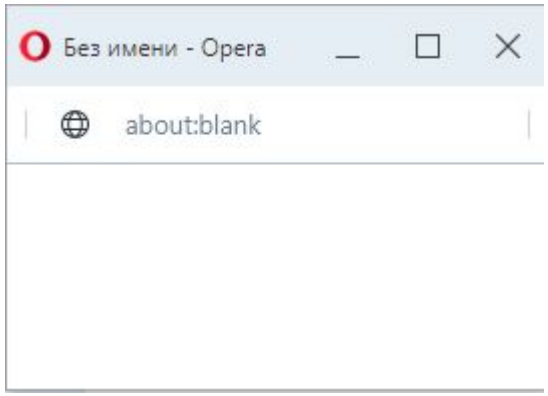
Параметр	Значення	Опис
width	число	Ширина вікна в пікселах (не меншь ніж 100)
height	число	Висота вікна в пікселах (не меншь ніж 100)
left	число	Відстань від лівого краю екрана до лівої межі вікна в пікселах
top	число	Відстань від верхнього краю екрана до верхньої межі вікна в пікселах
location	yes / no	Наявність у вікна поля адреси
menubar	yes / no	Наявність у вікна панелі меню
resizable	yes / no	Чи зможе користувач змінювати розмір вікна
scrollbars	yes / no	Наявність у вікна смуг прокрутки
status	yes / no	Наявність у вікна поля статусу
toolbar	yes / no	Наявність у вікна панелі інструментів

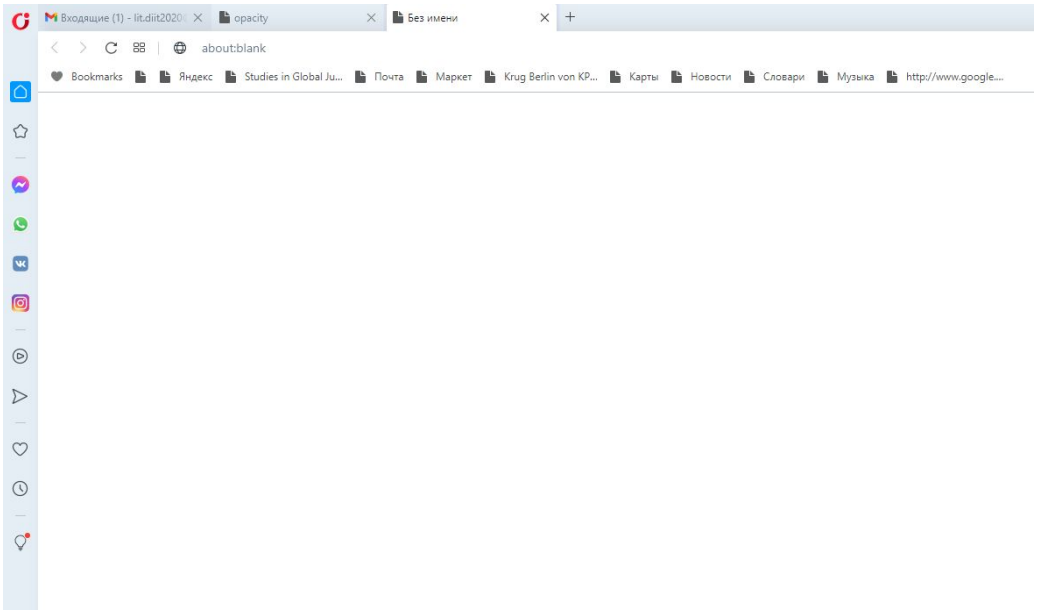
Методи об'єкта window

- Наведемо два приклади відкриття нового вікна:
- <form>
- <input type = button value = "просте вікно"
- onclick = "window.open ('', 'test1',
- 'directories = no, height = 200, location = no,' +
- 'menubar = no, resizable = no, scrollbars = no,' +
- 'status = no, toolbar = no, width = 200'); ">
-
- <input type = button value = "складне вікно"
- onclick = "window.open ('', 'test2',
- 'directories = yes, height = 200, location = yes,' +
- 'menubar = yes, resizable = yes, scrollbars = yes,' +
- 'status = yes, toolbar = yes, width = 200'); ">
- </ form>

Методи об'єкта window







Методи об'єкта window

- ***window.close ()*** – дозволяє закрити вікно. Найчастіше виникає питання, яке з вікон, власне, слід закрити. Якщо необхідно закрити поточний, то:
- window.close ();
- self.close ();

- Якщо необхідно закрити довільне вікно, то тоді спочатку потрібно отримати його ідентифікатор:
- id = window.open ();
- …
- id.close ();

Методи об'єкта window

- ***Метод focus ()*** застосовується для передачі фокусу у вікно, з яким він використовувався. Передача фокуса корисна як при відкритті вікна, так і при його закритті, не кажучи вже про випадки, коли потрібно вибирати вікна. Розглянемо приклад.
- Відкриваємо вікно і, не закриваючи його, знову відкриємо вікно з таким же ім'ям, але з іншим текстом. Нове вікно не з'явилося поверх основного вікна, так як фокус йому не був переданий.
- Щоб відвести фокус з певного вікна myWin, необхідно застосувати метод myWin.***blur ().***
- Наприклад, щоб відвести фокус з поточного вікна, де виконується скрипт, потрібно викликати window.blur (). Ефект буде той же, як якби користувач сам звернув вікно натисканням кнопки в правому верхньому куті вікна.

Події об'єкта window

- ***Load*** - подія відбувається в момент, коли завантаження документа в даному вікні повністю закінчилася.
- <body onload = "alert ('документ повністю завантажений.');">

- ***Unload*** - подія відбувається в момент вивантаження сторінки з вікна.
- Наприклад, коли користувач з нашої сторінки ми можемо подбати про його зручність і закрити відкрите раніше нашим скриптом вікно:
- <body onunload = "mywin.close ();">

- ***Error*** - подія відбувається при виникненні помилки в процесі завантаження сторінки. Якщо ця подія відбулася, можна, наприклад, вивести повідомлення користувачеві за допомогою alert () або спробувати перезавантажити сторінку за допомогою window.location.reload ().

Події об'єкта window

- ***Focus*** - подія відбувається в момент, коли вікна передається фокус. Наприклад, коли користувач "розкриває" згорнуте раніше вікно, або (в Windows) вибирає це вікно браузера за допомогою Alt + Tab серед вікон інших додатків. Ця подія відбувається також при програмної передачі фокусу даному вікна шляхом виклику методу window.focus (). Приклад використання:
- <body onfocus = "alert ('дякую, що знову повернулися!');">

- ***Blur*** - подія, протилежне попередньому, відбувається в момент, коли це вікно втрачає фокус. Це може статися в результаті дій користувача або програмними засобами - викликом методу window.blur ().

- ***Resize*** - подія відбувається при зміні розмірів вікна користувачем або сценарієм.

Об'єкт Document

- Об'єкт document є найважливішим властивістю об'єкта window
- До нього потрібно звертатися як
  - ***window.document….***

- Всі елементи HTML-розмітки, присутні на web-сторінці - текст, абзаци, гіперпосилання, картинки, списки, таблиці, форми і т.д. - ***є властивостями об'єкта document.***

- Можна сказати, що технологія DHTML (Dynamic HTML), тобто динамічна зміна вмісту web-сторінки, полягає саме в роботі з властивостями, методами і подіями об'єкта document (не рахуючи роботи з вікнами і фреймами).

Властивості, методи та події об'єкта document

Властивості	Методи	Події
URL Domain title last Modified Referrer cookie linkColor AlinkColor vlinkColor	open() close() write() writeln() getSelection() getElementById() getElementsByName() getElementsByTagName()	Load Unload Click DblClick MouseDown MouseUp KeyDown KeyUp KeyPress

Контейнер FORM

- Зазвичай контейнер FORM і поля форм іменовані:
- <form name = fname method = get>
- <input name = iname size = 30 maxlength = 30>
- </ form>

- Тому в програмах на JavaScript до них звертаються по імені:
- document.fname.iname.value = "Текст";

- Того ж ефекту можна досягти, використовуючи колекції форм і елементів, звертаючись до форми і до елементу або за індексом, або по імені:

- document.forms [0] .elements [0] .value = "Текст";
- document.forms ['fname']. elements ['iname']. value = "Текст";

Властивості, методи та події об'єкта Form

Властивості	Методи	Події
length action method target encoding elements[]	reset() submit()	reset submit

Властивості об'єкта Form

- ***Властивість action*** **-** відповідає за виклик CGI-скрипта. У ньому вказується URL цього скрипта. Але там, де можна вказати URL, можна вказати і його схему javascript :, наприклад:
- <form method = **post** action = "javascript: alert ('працює!');">
- <input type = submit value = "продемонструвати javascript в action">
- </ form>
- Метод POST передає дані форми скрипту в тілі HTTP-повідомлення, тому символ "?" Не додає до URL, і помилка не генерується. При цьому застосування void (0) скасовує перезавантаження документа, і браузер не генерує подію Submit, тобто не вдається до сервера при натисканні на кнопку, як це було б при стандартній обробці форми.

Властивості об'єкта Form

- ***Властивість method*** визначає метод доступу до ресурсів HTTP-сервера з програми-браузера. Залежно від того, як автор HTML-сторінки збирається отримувати і обробляти дані з форми, він може вибрати той чи інший метод доступу. На практиці найчастіше використовуються методи GET і POST.
- Зміна методу форми (GET і POST) скриптом
- <form name = f action = "javascript: alert ('працює!');">
- <script>
- document.write ('за замовчуванням встановлений метод:' + document.f.method + '.
');
- </ script>
- <input type = button onclick = "document.f.method = 'post'" value = "змінити метод на post">
- <input type = button onclick = "document.f.method = 'get'" value = "змінити метод на get">
- <input type = submit value = "javascript в action">
- </ form>

Властивості об'єкта Form

- ***Властивість target*** визначає ім'я вікна, в яке слід завантажувати результат звернення до CGI-скрипту.
- ***Колекція elements [] - п***ри генерації вбудованого в документ об'єкта Form браузер створює і пов'язаний з ним масив (колекцію) полів форми elements []. Зазвичай до полів звертаються по імені, але можна звертатися і за індексом масиву полів форми:

<form name = f>

<input name = e size = 40>

 <input type = button value = "ввести текст на ім'я елемента"

onclick = "**document.f.e.value** = 'текст в назві елемента';">

 <input type = button value = "текст за індексом елемента"

onclick = "**document.f.elements [0] .value** = 'текст введений за індексом елемента';">

 <input type = reset value = "очистити">

</ form>

Методи об'єкта Form

- ***Метод submit ()*** дозволяє проініціювати передачу введених в форму даних на сервер.
- <form name = f action = "http://www.intuit.ru/rating_students/">
- ваше ім'я користувача на intuit: <input name = query>
- </ form>
- переглянути рейтинг
- Кнопки відправки (submit) у форми немає, але натиснувши на посилання, ми виконуємо відправку даних на сервер.

- ***Метод reset ()*** дозволяє відновити значення полів форми, задані за замовчуванням.
- Іншими словами, виклик методу reset () рівносильний натискання на кнопку INPUT типу TYPE = reset, але при цьому ***кнопку створювати не потрібно.***

Події об'єкта Form

- ***Подія Submit*** виникає (і відповідний обробник події onSubmit викликається) при натисканні користувачем на кнопку типу submit або при виконанні методу submit ().
- Дія за замовчуванням, яку виконує браузер при виникненні цієї події - відправка введених в поля форми даних на сервер, вказаний в атрибуті ACTION, за допомогою методу, зазначеного в атрибуті METHOD, з використанням способу кодування, зазначеного в атрибуті ENCTYPE, і з зазначенням того, що результати роботи CGI-скрипта повинні бути показані у вікні або фреймі з ім'ям, зазначеним в атрибуті TARGET.
- ***Подія Reset*** виникає (і відповідний обробник події onReset викликається) при натисканні користувачем на кнопку типу reset або при виконанні методу reset ().
- Дія за замовчуванням, яку виконує браузер при виникненні цієї події - відновлення значень за замовчуванням в полях форми.

Поля форми та їх об'єкти

- <form name = f>
- <input type = text name = e value = "текст" onfocus = "">
- <input type = button name = b value = "кнопка" onclick = "">
- </ form>
- Тоді замість повного запису:
- <input type = text name = e value = "текст"
  onfocus = "alert (**document.f.e.value**)">
- ***АБО***
- <input type = text name = e value = "текст"
  onfocus = "alert (**value**)">
- В цьому контексті еквівалентні такі записи:
- value // коротше не буває!
- this.value // тут this посилається на елемент "e"
- form.e.value // form є властивість об'єкта "e" (рівне "f")
- this.form.e.value // комбінуємо обидва способи
- document.f.e.value // майже повний запис
- window.document.f.e.value // це найповніша запис
- document.f.e.form.e.value // можна ітерованих "form.e."

Текстове поле введення (об'єкт Text)

- Поля введення (контейнер INPUT типу TYPE = text) є одним з найбільш популярних об'єктів програмування на JavaScript.

- посилання 1
- <form> число гіпертекстових посилань до цього моменту:
- <script>
- document.write ('<input name = t value =' + document.links.length + '>');
- </ script>
-
 <input type = button
- value = "число посилань після закінчення завантаження сторінки"
- onclick = "form.t.value = document.links.length;">
-
 <input type = reset>
- </ form>
- посилання 2

Властивості, методи та події об'єкта Text

Властивості	Методи	Обробники подій	
defaultvalue value size maxlength disabled readonly	focus() blur() select()	onchange onselect onfocusonblur onclickondblclick	onmouseover onmouseout onmousedown onmouseup onkeypress onkeydown onkeyup

Списки варіантів (об'єкти SELECT і OPTION)

- Одним з важливих елементів інтерфейсу користувача є списки варіантів. У HTML-формах для їх реалізації використовується контейнер <SELECT>, який вміщує в себе контейнери <OPTION>. Залежно від наявності атрибута MULTIPLE у контейнера <SELECT> список може бути або з можливістю вибору тільки одного варіанту, або декількох варіантів.
- Для Select

Властивості	Методи	Обробники подій
options[] Size Length Multiple selectedIndex	focus() blur() add() remove()	onBlur onChange onFocus

- Option

Властивості	Методи	Обробники подій
defaultSelected selected index text value	нет	нет

Текстове поле введення (об'єкт Text)

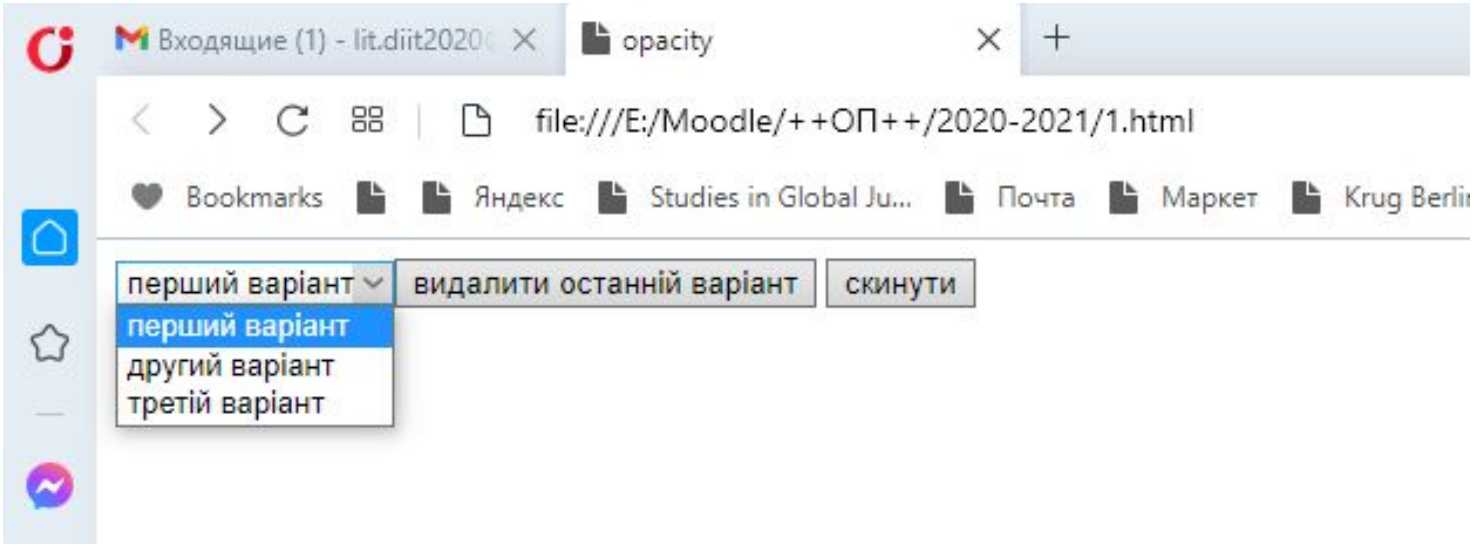
- ***Створення об'єктів Option***
- Об'єкт класу Option цікавий тим, що на відміну від багатьох інших вбудованих в DOM об'єктів JavaScript, має конструктор. Це означає, що програміст може сам створити об'єкт класу Option:
- **opt = new Option ([text, [value, [defaultSelected, [selected]]]]);**
- де аргументи відповідають властивостям звичайних об'єктів класу Option:

- text - рядок тексту, яка розміщується в контейнері <option> (наприклад: <option> текст </ option>);
- value - значення, яке передається серверу при виборі альтернативи, пов'язаної з об'єктом Option;
- defaultSelected - обрана ця альтернатива за замовчуванням (true / false);
- selected - альтернатива була обрана користувачем (true / false).

Колекція options []

- Вбудований масив (колекція) options [] - це одна з властивостей об'єкта Select. Елементи цього масиву є повноцінними об'єктами класу Option. Вони створюються в міру завантаження сторінки браузером.
- Кількість об'єктів Option, що містяться в об'єкті document.f.s класу Select, можна дізнатися за допомогою стандартного властивості масиву: document.f.s.options.length. Крім того, у самого об'єкта Select є таке ж властивість: document.f.s.length - воно повністю ідентично попередньому.
- <form>
- <select name = s>
- <option> перший варіант </ option>
- <option> другий варіант </ option>
- <option> третій варіант </ option>
- </ select>
- <input type = button value = "видалити останній варіант"
- onclick = "form.s.options [form.s.length-1] = null;">
- <input type = reset value = "скинути">
- </ form>

Колекція options []




```
function RestoreOptions ()
{
document.f.s.options [0] = new Option ( 'Варіант один', '', true, true);
document.f.s.options [1] = new Option ( 'Варіант два');
document.f.s.options [2] = new Option ( 'Варіант три');
return false;
}
</ script>
```

Кнопки

- У html-формах використовується чотири види кнопок:
- <form>
- <input type = button value = "кнопка типу button">
- <input type = submit value = "кнопка відправки">
- <input type = reset value = "кнопка скидання">
- <input type = image src = a.gif> <! - графічна кнопка ->
- </ form>

- Випадки в атрибуті кнопки або форми, задані користувачем:

1. при виклику методу click () кнопки викликається і обробник події onClick цієї форми;
2. при виклику методу submit () форми не викликається обробник події onSubmit форми;
3. при виклику методу reset () форми викликається і обробник події onReset форми.

Кнопки

- Кнопка типу ***button*** вводиться в форму головним чином для того, щоб можна було виконати будь-які дії або при її натисканні користувачем, або при виклику методу click ().
- <form name = f>
- <input type = button name = b value = "кнопка" onclick = "alert ('5 + 7 =' + (5 + 7))">
- </ form>
- викликати метод click ()
- Кнопка відправки (***submit***) дозволяє відправити дані, введені в форму, на сервер. У найпростішому випадку - при відсутності у контейнера <FORM> атрибутів ACTION (його значенням за замовчуванням є адреса поточної сторінки), METHOD (його значенням за замовчуванням є GET) і TARGET (його значенням за замовчуванням є _self) - стандартним дією браузера при відправці даних на сервер є просто перезавантаження поточної сторінки, що підтверджує наступний приклад:
- <form>
- <input type = submit>
- </ form>

Програмуємо графіку

- ***Об'єкт IMAGE***

Властивості	Методи	Події
Name Src lowSrc Border Height Width Hspace vspace complete	нет	Abort ErrorLoad

Об'єкт IMAGE

- Якщо є картинка (будемо вважати, що вона перша в документі):
- <img name = picname src = forest.gif>
- то значення властивості document.images [0] .name дорівнюватиме "picname", а до самої зображенні можна буде звертатися трьома способами:
- document.images [0]
- document.picname
- document.images ['picname']
- ***Властивості src і lowSrc***
- Властивості src і lowSrc визначають URL зображення, яке монтується всередину документа. При цьому lowSrc визначає тимчасове зображення, зазвичай маленьке, яке відображається, поки завантажується основне зображення, чий URL вказується в атрибуті SRC контейнера IMG. Властивість src приймає значення атрибута SRC контейнера IMG. Програміст може змінювати значення і src, і lowSrc. У попередньому прикладі ми можемо змінити значення src наступним чином:
- document.picname.src = 'river.gif';

Об'єкт IMAGE

- ***Зміна картинки***
- Змінити картинку можна, тільки присвоївши властивості src вбудованого об'єкта Image нове значення.
- Приклад (клікніть, щоб відкрити в новому вікні; для перегляду HTML-коду клікніть правою кнопкою миші і виберіть відповідний пункт)
- Рішення полягає в розведенні по часу підкачки картинки і її відображення. Для початку ми створюємо зображення, до яких прив'язуємо обробники подій onMouseOver і onMouseOut. При наведенні покажчика миші на кожну з картинок вона замінюється іншою (кольоровий), а при відведенні мишки картинка замінюється назад на чорно-білу:
- <img name = m0 src = "images / mapb000.gif" border = 0
- onmouseover = "document.m0.src = color [0] .src;"
- onmouseout = "document.m0.src = mono [0] .src;">

Оптимізація відображення

- При програмуванні графіки слід враховувати безліч факторів, які впливають на швидкість відображення сторінки і швидкість зміни графічних образів. При цьому звичайна дилема оптимізації програм - швидкість або кількість пам 'яті - вирішується тільки на користь збільшення швидкості. Про розмір пам'яті при програмуванні на JavaScript думати якось не прийнято.

- З усіх способів оптимізації відображення картинок ми зупинимося тільки на декількох:
- • оптимізація відображення при завантаженні;
- • оптимізація відображення за рахунок попереднього завантаження;
- • оптимізація відображення за рахунок нарізки зображення.

- Якщо перші дві позиції відносяться в рівній мірі як до відображення статичних картинок, так і до мультиплікації, то третій пункт характерний головним чином для мультиплікації.

Дякую за увагу!