Лабораторна робота № 1.

Основи JavaScript. Використання функцій в JavaScript. Події в Java Script

**Мета роботи**: набути навичок з написання скриптів та використання функцій на мові JavaScript.

***Теоретичні відомості***

JavaScript – мова програмування для створення скриптів, що розроблена фірмою Netscape. За допомогою JavaScript можна створювати інтерактивні Web-сторінки. Щоб запускати скрипти, що написані з використанням мови JavaScript необхідний браузер, що має змогу працювати з JavaScript. Код скрипта JavaScript розміщується безпосередньо на HTML-сторінці з використанням тегу <script>. Події та їх обробники є одними з важливих питань для програмування на мові JavaScript. Події, головним чином провокуються діями користувача. Наприклад, клік по кнопці або текстовому полю провокує виникнення події "Click". Якщо курсор миші перетинає будь-яке посилання гіпер-тексту – відбувається подія MouseOver та ін. Існує кілька різних типів подій, в даній лабораторній роботі ми розглянемо лише подію onClick. Наведений нижче код є прикладом простої програми обробки події onClick:

<form>

<input type="button" value="Click me" onClick="alert('Hello!!!')">

</form>

Однією з найвідоміших функцій є alert(), що дозволяє видати користувачеві інформацію або попередження у окремому вікні. При її виклику, у якості параметра передається деякий рядок, у випадку представленому нижче це “Hello!!!”. Таким чином, коли користувач клікає на кнопці або іншому елементі форми скрипт створює вікно, що містить текст ”Hello!!!”.

**Функції на мові JavaScript.**

У більшості програм на мові JavaScript як і у випадках інших мов програмування широко використовуються функції. У більшості випадках функції представляють собою лише поєднання кількох команд. Для прикладу розглянемо скрипт, що друкує деякий текст три рази поспіль:

<html>

<script language="JavaScript">

document.write("Ласкаво просимо на мою сторінку!<br>");

document.write("Це JavaScript!<br>");

document.write("Ласкаво просимо на мою сторінку!<br>");

document.write("Це JavaScript!<br>");

document.write("Ласкаво просимо на мою сторінку!<br>");

document.write("Це JavaScript!<br>");

</script>

</html>

Такий скрипт напише

Ласкаво просимо на мою сторінку!

Це JavaScript!

тричі. Як бачимо поставлена задача вирішена за допомогою того, що відповідна частина коду була дубльована три рази. Такий підхід не є ефективним. Наступний скрипт виконує ту ж саму задачу з використанням функції.

<html>

<script language="JavaScript">

function myFunction() {

document.write("Ласкаво просимо на мою сторінку!<br>");

document.write("Це JavaScript!<br>");}

myFunction();

myFunction();

myFunction();

</script>

</html>

І хоча кількість рядків у даному випадку не змінилась структура програми стала більш зрозумілою, читабельною, та придатною до подальшої модифікації. У даному програмному коді ми використали функцію, що містить наступні строки:

Отже, функції користувача — це функції, які ви можете створити самі, за власним вибором, для вирішення своїх задач. Функція задається своїм описанням, яке починається словом function і має наступний синтаксис:

function ім'я\_функції(параметри) {

код

}

Якщо функція повинна повернути деяке значення, то в її тілі використовується оператор return, справа від якого вказується те, що необхідно повернути. Цей оператор може міститись в тілі функції декілька разів.

Ініціалізація в тілі функції змінної з ключовим словом var створює локальну змінну, навіть якщо вона вже була оголошена у зовнішній програмі; якщо вона створена лише з допомогою оператора присвоювання, то вона є глобальною.

Приклад 4

function result(a,b)

{ c=a+b

return c

}

##### Видимість

 var b=1;

function fun1()

{

var a;

// тут існують змінні **а** та **b**

}

// тут існує лише змінна **b** (змінна **а** не існує )

## Оператори

#### Числові

+ Додавання для чисел та з'єднання для рядків

- Віднімання

\* Множення

/ Ділення

% Остача від ділення (12%5 повертає 2, УВАГА 12.3%5 повертає 2.3)

++ інкремент

-- декремент

### об'єкт Math

Об'єкт Math дозволяє виконувати різні математичні дії.

Об'єкт Math не є конструктором. Це означає, що всі властивості і методи об'єкту Math можуть викликані без його попереднього створення.

### синтаксис

var x = Math.PI; // Поверне число «пі»   
var y = Math.sqrt (16); // Поверне квадратний корінь числа 16

 .

### Властивості об'єкта Math

|  |  |
| --- | --- |
| **властивість** | **опис** |
| [**E**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_e.html&usg=ALkJrhgdMU3-EDX6WRknGI164SYOpnV2Hw) | Повертає число Ейлера (близько 2.718) |
| [**LN2**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_ln2.html&usg=ALkJrhgNSXoU5jZcWHyZ5vpYzS9H8qf1Og) | Повертає натуральний логарифм числа 2 (близько 0.693) |
| [**LN10**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_ln10.html&usg=ALkJrhhg3WlWlYXEecBvOoVsL_vuRSaNqg) | Повертає натуральний логарифм числа 10 (близько 2.302) |
| [**LOG2E**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_log2e.html&usg=ALkJrhgNFvR-mm908eyC8acwke7-aP9bsg) | Логарифм E за основою 2 (близько 1.442) |
| [**LOG10E**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_log10e.html&usg=ALkJrhjEFRrTdOo_wud_94WehS2OpUh4NQ) | Логарифм E по підставі 10 (близько 0.434) |
| [**PI**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_pi.html&usg=ALkJrhgsCGe_lb1fpXt2r7ZM_jf7g9TJoQ) | Число "пі" (близько 3.14159) |

### Методи об'єкта Math

|  |  |
| --- | --- |
| **методи** | **опис** |
| [**abs (x)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_abs.html&usg=ALkJrhjyCBQhMLcOyU1MrLWGGssQAyZw7Q) | Абсолютне значення числа x |
| [**ceil (x)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_ceil.html&usg=ALkJrhgzAvG4VwGm8qgrpC7wptdALNGOmA) | Округлення числа х в більшу сторону |
| [**floor (x)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_floor.html&usg=ALkJrhgqJuDxNSCLG_3pMJaLecSiPxmMtQ) | Округлення числа х в меншу сторону |
| [**max (x, y, z, ..., n)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_max.html&usg=ALkJrhj26xlBi1GuFY7-edP8WefJTSn_5g) | Обчислення максимуму з n чисел |
| [**min (x, y, z, ..., n)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_min.html&usg=ALkJrhh0IRP2xyNdRIuzKUHPUEpXXNsXJg) | Обчислення мінімуму з n чисел |
| [**pow (x, y)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_pow.html&usg=ALkJrhgtUk5e3LO17NUTqPdCLSKSoWILgQ) | Зводить число х в ступінь y |
| [**random ()**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_random.html&usg=ALkJrhgawpmCyDz62WKXjDK3M_pcveUDpg) | Повертає випадкове число від 0 до 1 |
| [**round (x)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_round.html&usg=ALkJrhjsXiHsJISDZF04qeZ3QTG6XlqmaQ) | Округлення чисел до найближчого цілого числа |
| [**sqrt (x)**](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://www.weblabla.ru/reference/javascript/jsref_sqrt.html&usg=ALkJrhiEXq4KeWmYThKUcs1kAYywJyz0dQ) | Обчислення квадратного кореня з числа х |

#### Присвоєння

a += b a = a + b

a -= b a = a - b

a \*= b a = a \* b

a /= b a = a / b

a %= b a = a % b

a &= b a = a & b

a |= b a = a | b

a ^= b a = a ^ b

#### Логічні

! NOT

&& AND (якщо перший операнд = false, другий операнд не обчислюється)

|| OR (якщо перший операнд = true, другий операнд не обчислюється)

#### Порівняння

== Equal

!= Not equal

> Greater

< Less

>= Greater or equal

<= Less or equal

<об'єкт>==<об'єкт> повертає true тільки якщо зліва й справа – один і той же об'єкт

null==undefined завжди true

Щоб порівняти <рядок>==<об'єкт>, < об'єкт > перетворюємо на рядок

У всіх інших випадках операнди перетворюємо на числа

#### Події

Сценарії в документі HTML призначені, зокрема, для обробки подій: натискання мишкою на елементі документа, наведення стрілки миші на елемент чи забирання її з нього, натиснення клавіші тощо. Більшість дескрипторів HTML мають спеціальні атрибути, що визначають події, на які можуть реагувати відповідні елементи. Список допустимих подій наведений нижче; він досить великий і розрахований на всі випадки життя. Значенням такого атрибуту-події є рядок, що містить сценарій — код-обробник події. Зазвичай обробники подій оформлюються у вигляді функцій, визначення яких розміщують у дескрипторі <script>. Розглянемо наступні приклади:

Приклад 1

<html>

<script>

function clickimage(){

alert ("Hello!")

}

</script>

<img src="pict.gif" onclick="clickimage()">

</html>

Приклад 2

<html>

<img src="morgan.jpg" onclick="alert('Hello!')" >

</html>

Є ще один метод, за допомогою якого можна визначати обробники подій. Майже для всіх дескрипторів HTML можна вказати атрибут ID — ідентифікатор. Його значенням є будь-який рядок, який грає роль індивідуального імені елемента в об’єктній моделі документа. З використанням цього атрибута для задання обробника події можна не використовувати атрибути-події; замість цього досить в контейнері <script> написати визначення функції-обробника події, ім'я якої створюється за шаблоном значення\_ID.подія(). Розглянемо наступний фрагмент коду:

Приклад 3

<html>

<h1 id= "Myheader">Привіт усім!</h1>

<script>

function Myheader.onclick(){

alert("Привіт")

}

</script>

</html>

Броузер, зустрічаючи в HTML-документі дескриптор з визначеним ідентифікатором ID, створює в об’єктній моделі цього документа об’єкт з тим же іменем. Для цього об’єкта існує метод обробки події. Назва метода співпадає з назвою події, однак синтаксис використання метода вимагає, щоб його ім'я було написане в нижньому регістрі. Елемент документа повинен бути завантажений раніше, ніж функція-обробник події. В таблиці 2.1 наведено основні події мови HTML.

Таблиця 2.1. Основні події для документів HTML

|  |  |  |  |
| --- | --- | --- | --- |
| Подія | До яких елементів застосовна | Коли відбувається | Обробник події |
| abort | Зображення | Користувач перериває завантаження, наприклад, натисканням на нове посилання чи кнопку зупинки | onabort |
| blur | Вікна, фрейми та всі елементи форм | Користувач переносить фокус введення з вікна, фрейму чи елемента форми | onblur |
| click | Всі кнопки, перемикачі, прапорці та посилання | Натискання мишею на елементі | onclick |
| change | Текстові поля та області, списки | Користувач змінює значення елемента | onchange |
| error | Зображення, вікна | Завантаження документа чи зображення призвело до помилки | onerror |
| focus | Вікна, фрейми та всі елементи форм | Користувач переносить фокус введення у вікно, фрейм чи елемент форми. | onfocus |
| load | Тіло документа | Користувач завантажує сторінку у броузер | onload |
| mouseout | Області, посилання | Користувач пересуває стрілку миші за межі області чи посилання | onmouseout |
| mouseover | Посилання | Користувач поміщає стрілку миші на посилання | onmouseover |
| reset | Форми | Користувач повертає форму у вихідний стан (натисканням кнопки Reset) | onreset |
| select | Текстові поля та області | Користувач виділяє поле вводу елемента форми | onselect |
| submit | Кнопки Submit | Користувач передає форму на сервер | onsubmit |
| unload | Тіло документа | Користувач закриває сторінку | onunload |
| keyup | Текстові поля та області | Користувач відпускає нажату клавішу | onkeyup |

**Об'єкт Event**

Зберігає інформацію про подію

ONCLICK="myHandler(event)".

**function reportEvent(ev) {  … }**

**document.links[2].onclick = reportEvent**

**type**    - тип події (наприклад, "click")

**target** – об'єкт, якому адресована подія

**layerX, layerY** – горизонтальна і вертикальна позиції курсора миші

**which** – номер натиснутої кнопки миші.

**modifiers**– чи натиснута клавіша-модифікатор (Alt, Ctrl, Shift). Бітова маска, одна з констант ALT\_MASK, CONTROL\_MASK, SHIFT\_MASK, and META\_MASK

**data** – масив з рядків, який містить URL об'єктів, які були перетягнуті та впущені у вікно.

**pageX, pageY**– горизонтальна та вертикальна позиції курсора відносно веб-сторінки

**screenX, screenY**– горизонтальна та вертикальна позиції курсора на екрані

**width, height**– ширина та висота вікна після зміни розмірів.

**Приклади**

**Відкрити/закрити вікно**

homeWindow=  open("bla.htm", "displayWindow",

    "width=400,height=300,status=no,toolbar=no,menubar=no");

homeWindow.close()

window.close()

self.close()

close()

**Випадаюче меню**

**Перевірка даних у формі**

**Динамічне завантаження елементів сторінки**

**Робота з сусіднім вікном**

**Новий документ в новому вікні**

*W= open("", "displayWindow",   "width=500,height=400,status=yes,toolbar=yes,menubar=yes");*

*W.document.open();*

*// новий документ*

*myWin.document.write("<html><head><title>On-the-fly");*

*…*

*myWin.document.write("</title></head><body>");*

*myWin.document.close();*

**Попереднє завантаження зображень**

hiddenImg= new Image();

hiddenImg.src= "img3.gif";

### Функції

В JavaScript є вбудовані функції, які можна використовувати в програмі, але їхній код не можна ні змінити, ні продивитись. Крім цих функцій, користувач може створювати свої власні .

#### Вбудовані функції

parseInt(рядок, основа) — перетворює вказаний рядок на ціле число у системі числення за вказаною основою (8, 10, 16); якщо основа не вказана, то за замовчуванням використовується 10.

parseFloat(рядок, основа) — перетворює вказаний рядок у число з плаваючою крапкою.

isNaN(значення) — повертає true, якщо вказане значення не є числом, і false в противному випадку.

eval(рядок) — обчислює вираз у вказаному рядку; вираз повинен бути написаний на мові JavaScript (не містити дескрипторів HTML).

escape(рядок) — повертає рядок у вигляді %XX, де ХХ — ASCII-код вказаного символу.

unescape(рядок) — здійснює обернене перетворення.

typeOf(об'єкт) — повертає тип вказаного об’єкта у вигляді символьного рядку.

prompt(text|[,value])— виводить повідомлення у вікні з текстовим полем і двома кнопками: "ОК" і "ВІДМІНА". Повертає введене значення або null, якщо відвідувач натиснув на кнопку "ВІДМІНА". Відвідувач не може робити нічого іншого, поки не вибере одну з кнопок.

Аргумен text:  текст повідомлення, value: рядок, введений в текстове поле за замовчуванням. Необов'язковий параметр.  
  
Приклад  
  
var years = prompt ('Скільки вам років?', 100)  
alert ('Вам' + years + 'років!')

Функції користувача

1. Оголошуємо функцію

<script>

function buttonOff() {  alert("Sorry, button disabled.") ; }

</script>

2. задаємо в коді HTML атрибут

 <INPUT id=btn TYPE="button" VALUE="Calculate" onClick=" buttonOff() ">

або в JavaScript

<script>

document.getElementById('btn').onclick = buttonOf

</script>

##### Видимість

 var b=1;

function a()

{

var a;

// тут існують змінні **а** та **b**

}

// тут існує лише змінна **b** (змінна **а** не існує )

### Об'єктна модель документа

Незалежно від базової стратегії розробки клієнта, всі підходи базуються на об’єктній моделі документа DOM (Document Object Model) — платформно-незалежному інтерфейсі броузера з відображуваним документом HTML. Специфікація такої моделі визначена консорціумом W3C (World Wide Web Consortium — W3C —це об’єднання розробників технологій та організацій, відповідальних за стандарти HTTP, HTML, XML, XSL, DOM та інші важливі стандарти Web та Internet), і більшість розробників броузерів реалізували її у останніх версіях своїх продуктів. Основна ідея полягає у використанні спільного інтерфейсу API, який розробники Web-сторінок можуть застосовувати для обробки вмісту документа HTML (або XML), а також ресурсів самого броузера.

При використанні об’єктної моделі документа програми та сценарії можуть динамічно отримувати доступ та оновлювати зміст документа, його структуру та стиль. Потім документ може оброблятись браузером, і результати цієї обробки можуть фіксуватись на відображуваній сторінці. За такої архітектури браузер відповідає як за відображення сторінки HTML, яка може бути змінена після її отримання з сервера, так і за виконання сценаріїв та компільованих програм в документі.

Назва моделі DOM пов’язана з тим, що вона забезпечує об’єктний інтерфейс до документів HTML (та XML). Документи розглядаються як об’єкти, що мають дані та поведінку. Взаємодія цих об’єктів представляє структуру документа.

### Об'єкти JavaScript

Об'єктна модель документа, головне джерело об’єктів для JavaScript, забезпечує об’єктний інтерфейс не лише для документів HTML, але й для броузера. Сценарій JavaScript може взаємодіяти з броузером для завантаження нової сторінки, перевіряти журнал броузера (список завантажених раніше Web-сторінок) або взаємодіяти з іншими сторінками у сусідніх фреймах.

Головним об'єктом при роботі з документами є document. Посилання на цей об'єкт можна отримати за допомогою атрибута цього об’єкта window, який є глобальним для будь-якої функції JavaScript. Для звернення до атрибутів в JavaScript використовується оператор „крапка” (.).

В об’єктній моделі документи згруповані у так звані колекції. Колекцію можна розглядати як проміжний об'єкт, що містить об’єкти власне документа. З іншого боку, колекція є масивом об’єктів, відсортованих у порядку слідування відповідних елементів в HTML-документі. Синтаксис звертання до елементів колекції такий же, як і для елементів масиву. Колекція має атрибут length — кількість всіх її документів. Колекція всіх елементів документа називається all; є також тематичні колекції: images, forms, links тощо (колекції усіх зображень, форм, посилань відповідно). Об'єкт може належати до якоїсь тематичної колекції, але обов’язково входить до колекції all. Загальні правила звертання до атрибутів всіх об’єктів такі:

document.колекція.id\_об’єкта

document.колекція["id\_об’єкта"]

document.колекція[індекс\_об’єкта]

### Об'єкти користувача JavaScript

Хоча JavaScript не є повноцінною об’єктно-орієнтованою мовою, вона забезпечує механізм для інкапсуляції. Інкапсуляція (encapsulation) — це можливість приховання даних всередині об’єкта. У мові JavaScript можна створити екземпляр родового об’єкта (generic object) і призначити йому атрибути і навіть методи. Наприклад, у наступному фрагменті коду на JavaScript створюється екземпляр родового об’єкту з іменем cartLineItem (один з товарів у віртуальному кошику покупця). За допомогою оператора „крапка” (.) задаються значення чотирьох вказаних користувачем властивостей.

Приклад 5

LineItem= new Object();

LineItem.productID = ‘MG1234’;

LineItem.productName = ‘MGB Roadster (1935)’;

LineItem.qty =1;

LineItem.unitPrice =36000;

Екземпляр LineItem можна використовувати в подальшому в будь-якому сценарії JavaScript з відповідним посиланням:

alert( ‘В вашому кошику є ’ + cartLineItem.qty + ‘ ’ + LineItem.productName);

Для об’єктів можна також визначати методи. Наприклад, наступну функцію total() можна використовувати як метод. Вона звертається до свого об’єкту за допомогою оператора this:

function total(){  
 return (this.qty \* this.unitPrice);  
}

Доступ до функції здійснюється таким же чином як і до атрибутів.

LineItem.total = total;

Тепер цю функцію можна викликати безпосередньо для об’єкта LineItem:

LineItem= new Object();

LineItem.productID = ‘MG1234’;

LineItem.productName = ‘MGB Mk I Roadster’;

LineItem.qty = 2;

LineItem.unitPrice = 12500;

LineItem.total = total;

document.write(‘<p>’ + LineItem.qty + ‘ ’ + LineItem.productName +‘стоит $’ + LineItem.total() + ‘</p>’);

Прототип в JavaScript нагадує конструктор в С++. Прототип (prototype) — це функція, що створює визначений користувачем об'єкт та ініціалізує його атрибути. Прототип для об’єкта JavaScript повинен також містити функції об’єкта. Прототип об’єкта cartLineItem повинен мати такий вигляд:.

function cartLineItem(id, name, qty, price){

this.productID = id;

this.productName = name;

this.qty = qty;

this.unitPrice = price;

this.total = total;

}

Якщо визначений прототип, то масив екземплярів LineItem можна створити за допомогою наступного фрагменту коду JavaScript:

var LineItem = new Array();

LineItem[0] = new cartLineItem (‘MG123’, ‘MGB Mk I Roadster’, 1, 36000);

LineItem[1] = new cartLineItem (‘AH736’, ‘Austin-Healey Sprite’, 1, 9560);

LineItem[2] = new cartLineItem (‘TS225’, ‘Triumph Spitfire Mk I’, 1, 11000);

#### Умовний оператор

**(a>b)?a : b** Якщо **a>b**, повернути **a**, інакше повернути **b**.

Cпеціальні оператори:

операция , (кома)

delete – знищити об'єкти або властивість об'єкта

new - створити новий об'єкт

this – звернення до поточного об'єкта

typeof – повертає тип аргумента

void - «не повертати значення»

## Умовні вирази

**Теоретичні відомості**: при розробці функціональної бази Інтернет сторінок з використанням мови JavaScript одним з необхідних елементів є умовні оператори. Виконання складних математичних операцій вимагає паралельної перевірки деяких умов. Структура умовного оператора в JavaScript має наступний вигляд:

if(<логічний\_вираз>) <оператор\_1> else <оператор\_2>

У даному випадку <логічний\_вираз> представляє собою повний або скорочений варіант певного виразу. При умові коли <логічний\_вираз> приймає значення true виконується <оператор\_1>, у іншому разі, тобто коли його значення дорівнює false <оператор\_2>. При побудові логічних виразів здебільшого застосовуються наступні операції порівняння:

|  |  |  |
| --- | --- | --- |
| № п/п | Позначення | Зміст |
| 1 | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_73acd28d.gif | дорівнює |
| 2 | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m2d599a89.gif | не дорівнює |
| 3 | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_1628fcdd.gif | більше |
| 4 | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m230ef85a.gif | більше або дорівнює |
| 5 | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m4291060b.gif | менше |
| 6 | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_5b0dc5b.gif | менше або дорівнює |

Слід зазначити, що у більшості випадках для скорочення програмного коду та підвищення рівня його читабельності використовуються наступні спрощення при записі логічних виразів:

|  |  |
| --- | --- |
| http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m5fbcefe2.gif | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_494fa0f3.gif |
| http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m11841036.gif | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_2370ea33.gif |
| http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_20f2168e.gif | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_21c21978.gif |
| http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m10c0af7d.gif | http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m6a5fc7d6.gif |

Можливі випадки коли при побудові логічних виразів необхідно врахувати не одну а кілька умов. Мова Java Script передбачає використання спеціальних символів побудови складних умовних операторів:

|  |  |
| --- | --- |
| Позначення | Оператор |
| http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_m67f90a3f.gif | і |
| http://www.studfiles.ru/html/2706/1145/html_HoZJ7_UJWj.yTQi/htmlconvd-qQkcCI_html_63dd561b.gif | або |

Структурний елемент else умовного оператору if у більшості випадках використовується для оптимізації програмного коду та удосконалення його читабельності. Слід зазначити, що у випадках коли варіанти перебору умовного оператору if є взаємно виключаючими, використання оператору else дозволяє скоротити об'єм перевірок і як результат пришвидшити роботу програми. У випадку коли варіанти перебору не доповнюють один одного використання структурного елементу else може спровокувати виникнення помилки. Нижче наведені два фрагменти коду, що демонструють не тільки оптимізаційні властивості умовних операторів, але й варіанти їх використання:

|  |  |
| --- | --- |
| Взаємовиключаючі умови | Взаємодоповнюючі умови |
| if(a==1) b=a; else  if(a==2) b=a\*a; else  if(a==3) b=a\*a\*a; else  if(a==4) b=1/a; else  if(a==5) b=1/a\*a; else | if(str.length) flag=1;  if(((str[2]==”a”)||(str[2]==”k”))&&(str[0]=” ”)) flag=0;  if(str.length>=5) flag=str.length  if(str==”twenty”) {number= 20; flag=0;} |

switch (expression){

case label :

statement;

break;

case label :

statement;

break;

...

default : statement;

}

**Цикли**

for (var i=0; i < 20; i++) {

…

}

do {

i+=1;

document.write(i);

}

while (i<5);

 n = 0

x = 0

while( n < 3 ) {

n ++

x += n

}

 for (var i in obj) { result += obj\_name + "." + i + " = " + obj[i] + "<BR>" }

**label** - оператор з ідентифікатором, що дозволяяє звернутися до нього в програмі.

markLoop: while (theMark == true) { …… }

##### break

для переривання циклу, або операторів switch або label.

1. break

2. break [label]

##### continue

continue - для рестарта циклу (while, do-while, for) або label.

checkiandj : while (i<4) {

checkj : while (j>4) {

…

if ((j%2)==0) continue checkj;

…

}

}

## Ітерація по всім властивостям об'єкта

for (var i in obj) { result += obj\_name + "." + i + " = " + obj[i] + "<BR>" }

## Коментар

<!-- Коментар -->