ចាប់ផ្តើម

Javascript & DOM Scripting

អ្នកនឹងសិក្សាពីមេរៀនតំបូង វិធីសាស្ត្រថ្មីៗ
និងវិធីសាស្ត្រប្រើជាញឹកញាប់ក្នុងការបង្កើតវេបសាយ

bayon hosting

រៀបរៀងដោយ: កុយ ទិត្យតារា

សំរាប់បង្រៀននៅ មជ្ឈមណ្ឌល ITEC និងចែកចាយតាម Bayon Hosting

Content

Module 1: Introduction to JavaScript

What is JavaScript?

JavaScript ជា client-side scripting language ដែលគេប្រើក្នុង Web Development ជាមួយនឹង ភាសា HTML, CSS ដើម្បីបង្កើត web page អោយមានលក្ខណៈរស់រវើក(Dynamic)។

Function of JavaScript

តួនាទីរបស់ JavaScript មានដូចជា៖

- បង្កើតនូវ message box, input box, confirm box
- validate នូវ Form Field ដូចជាការពិនិត្យ blank field
- ប្រើប្រាស់នូវ status bar
- ធ្វើការគណនា
- បង្កើតនូវរូបភាពមានចលនា
- បង្កើត drop down menu

JavaScript History

ផ្តើមចេញពីក្រុមហ៊ុន Netscape ដោយឃើញពីផលវិបាកដែល user ត្រូវរងចាំយ៉ាងយូរដោយ សារតែកំហុសបន្តិចបន្តួចរបស់ពួកគេ ដូចជាការ មិនបានវាយបញ្ចូល ការ វាយនូវអក្សរដែលមិន ត្រឹមត្រូវជាដើម។ រាល់ការ validate form ត្រូវធ្វើឡើងនៅលើម៉ាស៊ីន Server ដែលធ្វើអោយមាន ចរាចរណ៍ក្នុងបណ្តាញ Network ដែលនៅពេលនោះ ល្បឿនរបស់អ៊ីនធឺណិត មានត្រឹមតែ 28.8 kbps ប៉ុណ្ណោះ។

ក្រុមហ៊ុនបានបង្កើតនូវ client script មួយឈ្មោះថា live script ដើម្បីដោះស្រាយនូវបញ្ហា នេះ។ ក្រោយមក ក្រោមការសហការរបស់ក្រុមហ៊ុន Netscape ជាមួយនឹង Sun Microsystem ក្នុងការ អភិវឌ្ឍ បន្ថែមលើ live script បានប្តូរឈ្មោះទៅជា JavaScript វិញ។ ក្រោយមក JavaScript 1.0 ត្រូវ បានដាក់បញ្ចូលក្នុងកម្មវិធី Netscape Navigation 2.0 ក្នុងឆ្នាំ ១៩៩៥។

JavaScript 1.1 ត្រូវបានចេញក្នុង Netscape Navigation 3.0។ ក្នុងពេលនេះដែរ Microsoft ក៏ បានដាក់អោយប្រើនូវ Internet Browser តំបូងរបស់ខ្លួនឈ្មោះថា Internet Explorer 3.0។ ក្នុងជំនាន់ តំបូងនេះ Microsoft បង្កើតនូវ client script របស់ខ្លួនឈ្មោះថា JScript 1.0។

JavaScript Creation

យើងអាចបង្កើត javascript ដូចនឹង HTML អញ្ចឹង គឺគ្រាន់តែប្រើនូវ Text editor ដូចជា Notepad, Dreamweaver, Frontpage… ។

Internal JavaScript

គឺជាការដាក់កូដ JavaScript នៅក្នុង file តែមួយជាមួយគ្នានឹង file HTML ដោយប្រើនូវ <script/> tag ដើម្បីសំគាល់ពី javascript កូដ។

Syntax

```
<script type="text/javascript">
      statements;
</script>
```

External JavaScript

គឺជាដាក់កូដ javascript នៅ file ដាច់ដោយឡែកពី file HTML។ ដើម្បី link កូដ javascript មក html file ត្រូវប្រើនូវ <script/> tag ជាមួយនឹង src attribute ដោយកំនត់នូវទីតាំងរបស់ file javascript។ ជាធម្មតា file javascript ត្រូវបាន save ក្នុងទំរង់ extension .js ឧ. global.js។

Syntax

```
<script src="path"></script>
```

<noscript/> tag

ជា tag របស់ HTML ដែលប្រើសំរាប់ប្រើសំរាប់បង្ហាញពត៌មានអោយ User ដឹងថា Browser របស់ពួកគេមិន support រឺបាន disable javascript។

```
<noscript>
      Your Browser Not Support JavaScript or JavaScript was
disabled.
</noscript>
```

Module 2: JavaScript Basics

Generalization of JavaScript

លក្ខណៈមួយចំនួនរបស់ JavaScript មានដូចជា

- case-sensitive រាល់ឈ្មោះរបស់ variable គឺប្រកាន់អក្សរធំ រឺតូច។ ឧទាហរណ៍ variable ឈ្មោះថា student ខុសពី variable ឈ្មោះថា Student។
- រាល់ការប្រកាស variable មិនមានការបញ្ជាក់ពីប្រភេទទិន្នន័យរបស់ variable នោះទេ។
- ចុង statement យើងអាចដាក់ រឺមិនដាក់ semi-colon(;) បាន
- រាល់ប្លុកនៃកូដត្រូវដាក់ក្នុងសញ្ញា { } ដូចជា code block ក្នុង if statement, for statement, function...។

Comments

ជាការសរសេរពិពណ៌នាទៅលើកូដ javascript សំរាប់សំរួលដល់ចងចាំរបស់ programmer។ រាល់ comments មិនត្រូវបានបកប្រែដោយ browser ទេ។

Comments មានពីរប្រភេទគឺ៖

Single Line Comments

ជា Comment ដែលប្រើសំរាប់ពិពណ៌នាតែមួយបន្ទាត់ប៉ុណ្ណោះ

Syntax

```
// comments
```

Multiline Commnets

ជា Comment ដែលប្រើសំរាប់ការពិពណ៌នាច្រើនបន្ទាត់

Syntax

```
/* Comments
Comments
Comments
Comments */
```

Using Variables

Variable ជាអថេរដែលគេប្រើសំរាប់រក្សាទុកតំលៃបន្តោះអាសន្ន ណាមួយ រឺ reference ទៅកាន់Object ណាមួយ។

Declaration

ដើម្បីប្រកាស variable គេប្រើនូវ var statement។ ចំនាំថាយើងអាចប្រកាសដោយមិនចាំបាច់ផ្តល់តំលៃក៏បានដែរ។

syntax

```
var variableName;
var variableName=value;
```

យើងក៏ប្រកាសអថេរបានលើសពីមួយផងដែរក្នុង statement តែមួយដោយប្រើសញ្ញា commas(,) ដើម្បីខណ្ឌ variable នីមួយ។

```
var variable1,variable2,variable3;
var variable1=value1,variable2=value2;
```

Input into Variable

ដើម្បីបញ្ចូលតំលៃទៅអោយ variable យើងគ្រាន់តែប្រើនូវសញ្ញា assignement (=) និងបញ្ជាក់នូវឈ្មោះ variable និងតំលៃដែលត្រូវបញ្ចូល។

syntax

```
variable1=value1;
```

Output from Variable

ដើម្បីទាញយកតំលៃពី variable មួយយើងគ្រាន់បញ្ជាក់ឈ្មោះរបស់ variable ជាការស្រេច។

ឧទាហរណ៍៖

```
var sText = “Hello”;
alert(sText);
```

Naming Convention

- វាល់តួតំបូងរបស់ variable ត្រូវតែផ្តើមដោយអក្សរ, underscore(_), រឺ Dollar Sign($)។
- តួបន្តបន្ទាប់អាចជា អក្សរ, underscore(_), dollar sign($), រឺអាចជាលេខបាន
- វាល់ variable មិនត្រូវជាន់ជាមួយនឹង keyword រឺ reserved word របស់ JavaScript

Data Type

ប្រភេទទិន្នន័យគឺសំដៅ ទៅលើការប្រភេទនៃតំលៃ ដែរអថេរបានផ្ទុក។ ដើម្បីមើលប្រភេទទិន្នន័យនៃ variable មួយគេត្រូវប្រើនូវ keyword មួយឈ្មោះថា typeof ដែលមាន syntax ដូចខាងក្រោម៖

```
string typeof (variableName)
```

ប្រភេទទិន្នន័យសំខាន់ក្នុង JavaScript មានដូចជា៖

Undefined

ជាប្រភេទទិន្នន័យដែលតំនាងអោយ variable មួយដែលមិនទាន់បានបញ្ចូលតំលៃទៅអោយ។

Boolean

ជាប្រភេទទិន្នន័យដែលផ្ទុកតំលៃតែពីរប៉ុណ្ណោះគឺ true និង false ដែលត្រូវបានគេប្រើសំរាប់ការសិក្សាលក្ខខ័ណ្ឌ។

String

ជាប្រភេទទិន្នន័យជាអក្សរ ដែលមិនយកមកធ្វើការគណនាបានទេ។

Number

ជាប្រភេទទិន្នន័យជាលេខ ដែលយើងធ្វើការគណនាបាន ដូចជាការធ្វើប្រមាណវិធី បូក ដក គុណ ចែក ជាដើម។

Data Type Conversion

គេបំលែងប្រភេទទិន្នន័យមួយទៅ ប្រភេទទិន្នន័យដើម្បី ការគណនា ការប្រៀបធៀប។ ការបំលែងគឺមាន method សំរាប់ ប្រភេទទិន្នន័យនីមួយដូចខាងក្រោម៖

Number Conversion

គឺជាការបំលែងរាល់ប្រភេទទិន្នន័យផ្សេងៗអោយទៅជាប្រភេទទិន្នន័យជាលេខ៖

Table: Number Coversion

Value	`parseInt()`	`parseFloat()`	`Number()`	`eval()`
`false`	NaN	NaN	0	false
`true`	NaN	NaN	1	true
`undefined`	NaN	NaN	NaN	undefined
`"2.3"`	2	2.3	2.3	2.3
`"34"`	34	34	34	34
`"3+5"`	3	3	NaN	8

String Conversion

គឺជាការបំលែងរាល់ប្រភេទទិន្នន័យផ្សេងៗអោយទៅជាប្រភេទទិន្នន័យជាអក្សរ៖

Table: String Coversion

Value	`String()`	`.toString()`	`.toString(2)`	`.toString(8)`	`.toString(16)`
`false`	false	false	false	false	false

`true`	true	true	true	true	true
`undefined`	undefined	error			
`10`	10	10	1010	12	a
`14`	14	14	1110	16	e

Boolean Conversion

គឺជាការបំលែងប្រភេទទិន្នន័យដទៃទៀតអោយក្លាយជា Boolean។

Table: Boolean Conversion

Value	`Boolean()`
`""`	false
`"Letter"`	true
`undefined`	false
`0`	false
`14`	true
`oObject`	true

Operator

ជា symbol ដែលគេប្រើសំរាប់គណនា ប្រៀបធៀប និង ឈ្នាប់តក្ក: ។

Arithmetic Operator

គឺជាប្រមាណវិធីពីជគណិតសាមញ្ញដូចជា ការបូក ដក គុណ ចែក។

ឧបមាថាយើងមាននូវ variables មួយចំនួនដូចខាងក្រោម៖

```
var iNum1 = 9;

var iNum2 = 2;

var vResult;
```

Table: Arithmetic Operator

Operator	Meaning	Example	alert(vResult)
+	បូក	`iNum1 + iNum2`	11
-	ដក	`iNum1 - iNum2`	7
*****	គុណ	`iNum1 * iNum2`	18
/	ចែក	`iNum1 / iNum2`	4.5
%	ចែកយកសំណល់	`iNum1 % iNum2`	1
++	បូកបន្ថែមម្តងមួយ	`vResult=5;` `vResult++;`	5
		`vResult=5;` `++vResult;`	6
--	ដកចេញម្តងមួយ	`vResult=5` `vResult--;`	5
		`vResult=5;` `--vResult;`	4
+=	បូកបន្ថែមនឹងចំនួនណាមួយ	`vResult=5;` `vResult+=4;`	9
-=	ដកចេញនឹងចំនួនណាមួយ	`vResult=5;` `vResult-=4;`	1

Comparative Operator

គឺជា operator ដែលប្រើសំរាប់ការប្រៀបធៀប។ រាល់ការប្រៀបធៀបវាតែងតែបោះតំលៃជា boolean សំរាប់ការពិនិត្យលក្ខខ័ណ្ឌ

ឧបមាថាយើងមានន្ូវ variables មួយចំនួនដូចខាងក្រោម៖

```
var iNum1 = 9;
var iNum2 = 2;
var vResult;
```

Table: Comparative Operator

Operator	Meaning	expression	alert(expression)
==	ស្មើ (equal)	`iNum1 == iNum2`	`false`
!=	ខុសពី	`iNum1 != iNum2`	`true`
<	តូចជាង	`iNum1 < iNum2`	`false`
<=	តូចជាឬស្មើ	`iNum1 <= iNum2`	`false`
>	ធំជាង	`iNum1 > iNum2`	`true`
>=	ធំជាងឬស្មើ	`iNum1 >= iNum1`	`true`

Logical Operator

ជាឈ្មាប់ដែលគេប្រើសំរាប់ តក្កវិជ្ជាដែលមានតំលៃតែ true និង false ប៉ុណ្ណោះ។ យើងដឹងថា ឈ្មាប់និង(&&) ពិតលុះត្រាតែ operand ទាំងសងខាង ពិតទាំងពីរ ចំនែកឯ ឈ្មាប់រឺ(||) ពិតក្នុងករណីដែល operand មួយណាពិត។ ឈ្មាប់មិនគឺបញ្ច្រាស់ពី oprand បើ operand true វានឹងក្លាយជា false។

ឧបមាថាយើងមាននូវ variables មួយចំនួនដូចខាងក្រោម៖

```
var iNum1 = 9;
var iNum2 = 2;
var vResult;
```

Table: Logical Operator

Operator	Meaning	expression	alert(expression)
&&	ឈ្មាប់និង	`(iNum1 > iNum2) && (iNum1 != iNum2)`	`true`
||	ឈ្មាប់រឺ	`(iNum1 < iNum2) || (iNum1 = iNum2)`	`false`
!	ឈ្មាប់ខុសពី	`!(iNum1<iNum2)`	`true`

Control Flow Statements

If Statements

ជា statement ដែលបានប្រើសំរាប់ពិនិត្យមើលលក្ខខណ្ឌ បើលក្ខខណ្ឌពិត វានឹងតំណើរការន្ធវ statements របស់វា តែបើមិនពិតវានឹងមិនតំណើរការកូដរបស់វាទេ។

Syntax

```
if(expression){
}
```

Syntax

```
if(expression){
      statements;
}else if(expression){
      statements;
}else if(expression){
      statements;
}else{
      statements;
}
```

Switch Statements

វាវាយតំលៃលើ expression មួយ ហើយលោតទៅកាន់ statement ដែលបាន label ជាមួយ case clause ដែលត្រូវគ្នានឹងតំលៃរបស់ expression នោះ។ បើគ្មាន case ណាមួយត្រូវគ្នាទេ នោះ switch statement នឹងលោតទៅកាន់ statement ដែល label ដោយ default។

Syntax:

```
switch(expression){
      case value1: statement1;
            break;
      case value2: statement2;
            break;
      ...
      case valueN: statementn;
            break;
      default: default_statement
}
```

expression: ជា variable expression ដែលត្រូវពិភាក្សា

value: ជាតំលៃថេរដែលត្រូវផ្ទៀងផ្ទាត់ នឹង expression

break: ប្រើសំរាប់បិទការ execute របស់ statement

Iterative or Loop Statements

While Statement

ជា pre-test loop ដែលមានន័យថាវាធ្វើការពិនិត្យលើលក្ខខ័ណ្ឌមុននឹង execute code. វាធ្វើការ execute statements ម្តងហើយម្តងទៀត កាលណា expression មានតំលៃ true.

```
syntax:
while(expression){
      statements;
}
```

ឧទាហរណ៍៖

```
var i=0;
while(i<10){
      document.write(" Number " + i + "<br/>")
      i++;
}
```

Do While Statement

ជា post-loop ដែលមានន័យថាវាធ្វើការ excecute code មុន ហើយពិនិត្យលក្ខខ័ណ្ឌជាក្រោយមានន័យថា statements ត្រូវបាន execute យ៉ាងហោចណាស់ក៏បានម្តងដែរ។

```
syntax:
do{
      statements;
} while(expression);
```

ឧទាហរណ៍ ១៖

```
var i=0;
do{
      document.write(" Number " + i + "<br/>");
      i++;
}
```

For Statement

ដូចទៅនឹង while ដែរ ខុសត្រង់ថាវា សំរួលដល់ការបង្កើត initialization, condition, និង increment statement។

Syntax:

```
for(initialization,condition,increment_statement){
      statements;
}
```

ឧទាហរណ៍៖

```
for(i=1;i<=31;i++){
      document.write("Number : "+ i);
}
```

Break, Continue and Label Implementation

Break and Continue

Break

ជា statement ដែលប្រើសំរាប់ បញ្ឈោអោយចាកចេញពី loop។

ឧទាហរណ៍៖

```
for(i=0;i<10;i++){
      if(i==5)break;
      document.write("Number : "+ "<br/>");
}
```

Continue

ជា statement ដែលប្រើសំរាប់ បញ្ឈោអោយ loop បន្តជុំបន្ទាប់ ។

ឧទាហរណ៍៖

```
for(i=0;i<10;i++){
      if(i==5)continue;
      document.write("Number : "+ "<br/>");
}
```

Label

ប្រើសំរាប់ពេលយើងប្រើ loop ច្រើនជាន់។ វាធ្វើការអោយឈ្មោះទៅកាន់ loop ដូច្នេះ យើងអាចយើងប្រើនូវ break រឺ continue ដើម្បីបញ្ជារ loop ណាមួយបាន។

```
Syntax
      label_Name:
      loop
```

Functions

ជាបណ្តុំនៃ statements ដែលយើងអាចហៅវាមកប្រើបានគ្រប់ពេលទាំងអស់។ function ត្រូវបានប្រកាសជាមួយនឹង keyword `function`, ជាមួយនឹបណ្តុំនៃ arguments និង កូដ ដែលត្រូវដំនើរការក្នុង `{}`

Syntax:

```
function functionName(){

      statements

}
```

ឧទាហរណ៍៖

```
function showMessage(){

      alert("Hello world");

}
```

Using Arguments

គេប្រើ arguments សំរាប់បញ្ជូនតំលៃពីក្រៅ function ចូលទៅក្នុង functions ដើម្បីធ្វើអោយ function មានលក្ខណៈ dynamic។ ដើម្បីបង្កើត argument គេត្រូវដាក់ ឈ្មោះ varaibles ក្នុង កន្សោម arguments។

Syntax

```
function functionName(arg0,arg1,.....,argN){

      statements;

}
```

ឧទាហរណ៍៖

```
function showMessage(sMessage){

      alert(sMessage);
```

```
}
showMessage("Welcome to my Site !");
```

Using return

គេប្រើ return ដើម្បីបោះតំលៃចេញពី function វិញពេលគេហៅវាទៅប្រើ។

Syntax

```
function functionName(arg0,arg1,.....,argN){
      statements;
      return value;
}
```

ឧទាហរណ៍៖

```
function sum(fOperand1,fOperand2){
      return (fOperand1+fOperand2);
}
showMessage("Welcome to my Site !");
```

Module 3: Array and String

Array

Array ត្រូវបានគេកំនត់ថាជាបណ្តុំនៃទិន្នន័យដែលមានលក្ខណៈរួមគ្នាដូចជា ពណ៌ ឈ្មោះខែ ឈ្មោះថ្ងៃ។ គេប្រើ Array ដើម្បីសំរួលដល់ផ្ទុកទិន្នន័យ និងការប្រើប្រាស់ loop ដោយធាតុនីមួយៗរបស់ array ត្រូវតំនាងអោយលេខ index ។ លេខ index របស់ array ចាប់ពីផ្តើមពីលេខ 0 ទៅ។

Creating an Array

ដើម្បីបង្កើត array មួយ គេមានវិធីបីយ៉ាងដូចខាងក្រោម៖

Empty Array

ជាការបង្កើតនូវ array មួយដែលមិនទាន់មានធាតុ។ គេអាចប្រើនូវ new Array() រឺ សញ្ញា [ ]។

Example

```
var arrStudent = new Array();
var arrColor = [];
```

Specified number of items in Array

ដើម្បីបង្កើត array ដែលមានចំនួនធាតុជាក់លាក់ ជាស្រេចគេត្រូវបញ្ជាក់នូវចំនួនធាតុ new Array(n)។

Example

```
var arrStudent = new Array(39);//there are 39 items in array
```

Specified items in Array

ជាការបញ្ចូលតំលៃទៅក្នុង array ពេលប្រកាស់តែម្តង។ ដើម្បីធ្វើបែបនេះយើងត្រូវដាក់នីមួយៗបន្តបន្ទាប់គ្នាដោយខណ្ឌដោយសញ្ញា (,) ដូចខាងក្រោម៖

```
var arrColor = new Array("Blue","Black","Red","Yellow");
var arrFruit = ["Apple","Book","Car","Door","Eye"];
```

Input and Output from an array

ដើម្បីទាញយក រឺក៏បញ្ចូលធាតុទៅ array object យើងត្រូវប្រើ index ដើបញ្ជាក់ពីទីតាំង របស់ធាតុក្នុង array object នោះ។ Index ត្រូវបានគេចាប់គិត ពី 0 ទៅ ដូចនេះដើម្បីទាញយក ធាតុទី១នៃ array យើងត្រូវប្រើ index 0 ជំនួសវិញ។

ឧទាហរណ៍៖

```
var oStudent=new Array("Dara","Kagha","Yarern");
```

```
alert(oStudent[0]); // Dara

oStudent[0]=”Thea”;
alert(oStudent[0]);// Thea
```

ចំណាំ៖

Array អាចផ្ទុកធាតុបានច្រើនបំផុត 4294967295។

Array មាន Property តែមួយគត់គឺ `length` ដែលប្រើសំរាប់រាប់ចំនួនធាតុដែលមានក្នុង array object។

ឧទាហរណ៍៖

```
var arrStudent = new Array(15);
alert(oArray.length);// Output: 15

oInfo=[2,”Thera”,”Male”];
alert(oArray.length); //output: 3
```

Table: Array Methods

Methods	**ការពិពណ៌នា**
array concat(*value, ...*) **array concat(*array, ...*)**	បោះនូវ array ដែលមានផ្ទុកនូវធាតុរបស់ ចាស់ បន្ថែមដោយ value រឺ array។
string join(*separator*)	បោះនូវ String មួយដែលផ្ទុកនូវ គ្រប់ធាតុរបស់ array ដោយខ័ណ្ឌធាតុនីមួយៗដោយ separator ដែលជា string។
variant pop( )	លុបនូវធាតុដែលមានទីតាំងក្រោយបំផុតរបស់array ហើយបោះតំលៃនៃធាតុដែលបានលុបនោះ
number push(*value, ...*)	បន្ថែម value ថ្មីទៅកាន់ទីតាំងថ្មីក្រោយបំផុតនៃ array ហើយបោះនូវចំនួនធាតុទាំងអស់ក្នុងនោះ។
void reverse( )	ធ្វើការត្រឡប់ទីតាំងធាតុទាំងអស់ក្នុង array។
variant shift( )	លុបនូវធាតុដែលមានទីតាំងតំបូងបំផុតរបស់array ហើយបោះតំលៃនៃធាតុដែលបានលុបនោះ
array slice(*start, end*)	បោះនូវ array ថ្មីមួយដែលមានធាតុចាប់ពីទីតាំងstart ដល់ទីតាំងនៃ end-1.
void sort()	ប្រើសំរាប់តំរៀបធាតុក្នុង array តាមលំដាប់នៃ ASCII Code
array splice(*start, deleteCount, value,...*)	លុបនូវចំនួន deleteCount ចេញពី array ដោយគិតពីទីតាំង start និង បន្ថែមធាតុ values ចូលក្នុង ទីតាំង start បន្ត។ វាបោះ array ដែលមានផ្ទុកធាតុដែលបានលុបចេញ។
string toLocaleString( )	បោះនូវ String ដែលបានពីការបន្តធាតុនីមួយៗ ដោយ String ដែលកំនត់ដោយម៉ាស៊ីន។

string toString( )	បោះនូវ String ដែលបានពីការបន្តធាតុនីមួយៗ ដោយ (,)។
variant unshift(*value, ...*)	បន្ថែម value ថ្មីទៅកាន់ទីតាំងតំបូងបំផុតនៃ array ហើយបោះនូវចំនួនធាតុទាំងអស់ក្នុងនោះ។

String

String ជាប្រភេទទិន្នន័យដែលផ្ទុកទិន្នន័យដែលមានលក្ខណៈជាអក្សរ។ គេមាននូវ method ជាច្រើនដើម្បីធ្វើការទៅលើ String ដូចជាការ search, replace, ធ្វើការជាមួយ ascii code, បំលែង case។

Property in String

String ក៏ដូច array ដែរវាមាន property តែមួយគត់គឺ length ដែលប្រើសំរាប់ទាញយកចំនួនតួអក្សរដែលមានក្នុង string មួយ។

```
var sText = "Hello";
alert(sText);//Ouput 5
```

Methods in String

String មាននូវ methods ជាច្រើនដូចខាងក្រោម៖

Table: String Methods

Methods	ការពិពណ៌នា
string charAt(*n*)	បោះ អក្សរ ក្នុងទីតាំង n
number charCodeAt(*n*)	បោះនូវ ASCII Code របស់អក្សរ ក្នុងទីតាំង n
string concat(*value, ...*)	បោះនូវតំលៃនៃ ការបន្ត នូវ String ដើម នឹង value
number indexOf(*substring[, start]*)	បោះនូវទីតាំងតំបូងនៃ substring ក្នុង String ដើម start ជាចំនុចចាប់ផ្តើមស្វែងរក(0 Default)។បោះនូវតំលៃ -1 កាលណាមិនមាន។រាប់ពី0ទៅ
number lastIndexOf(*substring[, start]*)	បោះនូវទីតាំងក្រោយបង្អស់នៃ substring ក្នុង String ដើម។ start ជាចំនុចចាប់ផ្តើមស្វែងរក (length is Default)។បោះនូវតំលៃ -1 កាលណាមិនមាន។ រាប់ពីចុងមកទីតាំង0
string slice(*start[, end]*)	បោះនូវ String ដែលមានផ្ទុកនូវ string ពីទីតាំង `start` ដល់ `end`(មិនយក)។ (-) រាប់ពីចុងមកវិញ។ បើមិនមាន `end` វាគិតដល់ទីតាំងចុង។

array split(*delimiter[, limit]*)	បោះនូវ Array ដែលមានផ្ទុកនូវតំលៃធាតុដែលខណ្ឌាដោយ `delimiter`។
string substring(*from, to*)	បោះនូវ String ដែលមានផ្ទុកនូវ string ពីទីតាំង `from` ដល់ `to`(មិនយក)។ (-) មិនអនុញ្ញាតិ។ បើមិនមាន `to` វាគិតដល់ទីតាំងចុង។
substr(*start, length*)	បោះនូវ String ដែលមានផ្ទុកនូវ string ពីទីតាំង `start` ចំនួន `length`។ បើមិនមាន `length` វា គិតដល់ទីតាំងចុង។
toLowerCase( )	បោះនូវ String ដែលមានផ្ទុកនូវ string ដែលបំលែងទៅជា lowercase។
toUpperCase( )	បោះនូវ String ដែលមានផ្ទុកនូវ string ដែលបំលែងទៅជា uppercase។

Module 4: Date and Math

Date Object

ជាការធ្វើការជាមួយ ពេលវេលា ដូចការ មើល ពេលវេលាបច្ចុប្បន្ន, ធ្វើការគណនា រយៈពេល។

Creating a Date Object

សំរាប់ពេលវេលា បច្ចុប្បន្ន

```
var oNow = new Date( );
```

សំរាប់ពេលណាមួយដោយប្រើ timestamp

```
var oDate = new Date(timestamp);
```

ចំណាំ៖ Timestamp ជាចំនួនមីលីវិនាទីគិតចាប់ពីឆ្នាំ ១៩៧០ ខែ មករា ថ្ងៃទី ១ ម៉ោង ០ គត់ មកដល់ពេល បច្ចុប្បន្ន។

សំរាប់ពេលណាមួយដោយប្រើ datestring

```
new Date(datestring);
```

ចំណាំ៖ ជាប្រភេទពេលវេលាដែល javaScript ស្គាល់ដូចជា៖

```
Thu Sep 11 2008
Thu, 11 Sep 2008 10:24:39 UTC
Thursday, September 11, 2008
Thursday, September 11, 2008 5:24:39 PM
5:24:39 PM
Thu Sep 11 17:24:39 UTC+0700 2008
17:24:39 UTC+0700
Thu, 11 Sep 2008 10:24:39 UTC
```

សំរាប់ពេលណាមួយដោយប្រើ argument ជាក់លាក់

```
new Date(year, month, day, hours, minutes, seconds, ms);
```

ចំណាំ៖ Range នៃ argument នីមួយៗមានដូចខាងក្រោម៖

```
Year            :       4-digit
Month           :       0-11
Day             :       1-31
Hours           :       0-23
Minutes         :       0-59
Seconds         :       0-59
Ms              :       0-999
```

Methods in date object

ចំណាំ៖ UTC = Universal Time Coordinate

GMT= Greenwich Mean Time

ចំពោះ methods ណាដែលមាន [UTC] នោះមានន័យ យើងអាចដាក់ក៏បាន មិនបាច់ដាក់ក៏បានដែរ។ បើយើងដាក់ មានន័យថាយើងធ្វើការលើម៉ោងសាកល។

Table: Date Methods

Methods	ការពិពណ៌នា
number get[UTC]Date( )	បោះថ្ងៃ នៃខែពី ១ដល់ ៣១
number get[UTC]Day( )	បោះថ្ងៃ នៃសប្តាហ៍ពី ០ដល់ ៦
number get[UTC]FullYear( )	បោះផ្នែកឆ្នាំ ចំនួន ៤ខ្ទង់
number get[UTC]Hours( )	បោះផ្នែ កម៉ោងពី ០ ដល់ ២៣
number get[UTC]Milliseconds( )	បោះផ្នែក វិនាទីពី ០ ដល់ ៩៩៩
number get[UTC]Minutes( )	បោះផ្នែក នាទីពី ០ ដល់ ៥៩
number get[UTC]Month( )	បោះផ្នែក ខែពី ០(មករា) ដល់ ១១(ធ្នូ)
number get[UTC]Seconds( )	បោះផ្នែក វិនាទីពី ០ ដល់ ៥៩
number getTime( )	បោះចំនួនដែលបានមកពីការ រាប់ចំនួន មីលីវិនាទីពី ឆ្នាំ ១៩៧០ ខែមករា ថ្ងៃទី ១ ម៉ោង ០ គត់នៃម៉ោងសាកល(UTC) ដល់ពេល ដែលបានកំនត់ក្នុងDate Object។
number getTimezoneOffset( )	បោះចំនួននាទី ដែលបានមកពីការ យកម៉ោងនៅ Greenwich មកដកនឹង ម៉ោងម៉ាស៊ីន។
number getYear( )	ដូចនឹង getFullYear() ខុសត្រង់ថា ក្នុងចន្លោះ ១៩០០ដល់១៩៩៩ វាបោះតំលៃតែ ២ ខ្ទង់ប៉ុណ្ណោះ។
number set[UTC]Date(*day_of_month*)	កំនត់ខែអោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។
number set[UTC]FullYear(*year*, *[month*, **number *day]*)**	កំនត់ឆ្នាំ អោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។
number set[UTC]Hours(*hours[*, *mins*, **number *secs*, *ms]*)**	កំនត់ម៉ោង អោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។
number set[UTC]Milliseconds(*millis*)	កំនត់វិនាទី អោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។
number set[UTC]Minutes(*minutes*, **number *[seconds*, *millis]*)**	កំនត់នាទី អោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។
number set[UTC]Month(*month*, *[day]*)	កំនត់ខែអោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។

number set[UTC]Seconds(*seconds*, number *[millis]*)	កំនត់ វិនាទីអោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។
number setTime(*milliseconds*)	កំនត់ timestamp អោយ Date Object ហើយបោះនូវ timestamp នៃ Date ថ្មីនោះ។
number setYear(*year*)	ដូចនឹង setFullYear() ដែរខុសត្រង់ថា វាកំនត់តែ ២ខ្ទង់សំរាប់ ឆ្នាំ ១៩០០-១៩៩៩
string toDateString( )	បំលែង Date object ទៅទំរង់ ៖ Thu Sep 11 2008
string toGMTString( )	បំលែង Date object ទៅទំរង់ ៖ Thu, 11 Sep 2008 10:24:39 UTC
string toLocaleDateString( )	បំលែង Date object ទៅទំរង់ ៖ Thursday, September 11, 2008
string toLocaleString( )	បំលែង Date object ទៅទំរង់ ៖ Thursday, September 11, 2008 5:24:39 PM
string toLocaleTimeString( )	បំលែង Date object ទៅទំរង់ ៖ 5:24:39 PM
string toString( )	បំលែង Date object ទៅទំរង់ ៖ Thu Sep 11 17:24:39 UTC+0700 2008
string toTimeString( )	បំលែង Date object ទៅទំរង់ ៖ 17:24:39 UTC+0700
string toUTCString( )	បំលែង Date object ទៅទំរង់ ៖ Thu, 11 Sep 2008 10:24:39 UTC
number valueOf( )	ដូចនឹង getTime()

Static Method

Date ក៏មាននូវ methods ចំនួនពីរដែលយើងអាចយកមកប្រើ ដោយមិនចាំបានបង្កើត instance អោយ date class នោះទេ។

Table: Static Methods

Methods	ការពិពណ៌នា
number Date.parse(*datestring*)	បំលែងពីdatestring ទៅជា timestamp
number Date.UTC(*yr*, *mon*, *day*, *hr*, *min*, *sec*, *ms*)	បោះនូវ timestamp នៃ arguments ទាំងនោះ

Math Object

ជា Object ដែលមានន្ទវ methods និង properties សំខាន់ៗជាច្រើនសំរាប់ ប្រើក្នុងការគណនា រឹសកាវេ, ត្រីកោណមាត្រ និង យកតំលៃថេរដូចជា e, π ។

Math Properties

Table: Math Properties

Property	ការពិពណ៌នា
`Math.E`	តំលៃថេរ ស្មើនឹង 2.718.
`Math.LN10`	តំលៃថេរ នៃលោការីតនៃ១០.
`Math.LN2`	តំលៃថេរ នៃលោការីត នៃ ២.
`Math.LOG10E`	តំលៃថេរ នៃលោការីតគោល ១០ នៃ e.
`Math.LOG2E`	តំលៃថេរ នៃលោការីតគោល ២ នៃ e.
`Math.PI`	តំលៃថេរ នៃតំលៃ π
`Math.SQRT1_2`	តំលៃនៃ $1/\sqrt{2}$
`Math.SQRT2`	តំលៃនៃ $\sqrt{2}$

Math Methods

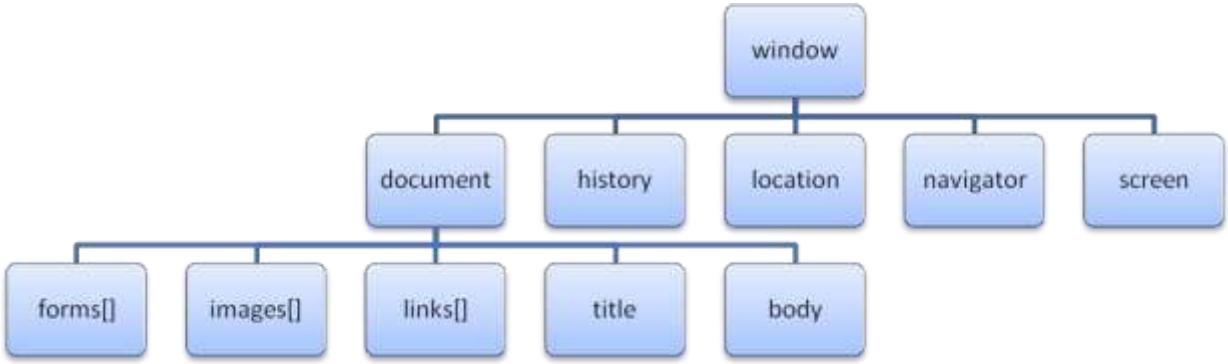
Table: Math Methods

Methods	ការពិពណ៌នា
`Math.abs( x)`	តំលៃដាច់ខាតនៃ x
`Math.acos( x)`	បោះតំលៃ arc cosine នៃ x
`Math.asin( x)`	បោះតំលៃ arc sine នៃ x
`Math.atan( x)`	បោះតំលៃ arc tangent នៃ x
`Math.atan2( y, x)`	បោះតំលៃ arc cosine នៃ y/x
`Math.ceil( x)`	បោះតំលៃធំជាងបន្ទាប់នៃ x
`Math.cos( x)`	បោះតំលៃ cosine នៃ x
`Math.exp( x)`	បោះតំលៃ e^x
`Math.floor( x)`	បោះតំលៃតូចជាងបន្ទាប់នៃ x
`Math.log( x)`	បោះតំលៃ log(x)
`Math.max( args...)`	បោះតំលៃធំជាងគេក្នុងចំនោម args ទាំងអស់

Math.min(*args...*)	បោះតំលៃតូចជាងគេក្នុងចំនោម args ទាំងអស់
Math.pow(*x*, *y*)	បោះតំលៃ x^y
Math.random( )	បោះតំលៃលេខចន្លោះពី 0.0 ដល់ 1.0
Math.round(*x*)	បោះតំលៃដែលបង្គត់ពីតំលៃ x
Math.sin(*x*)	បោះតំលៃ sine នៃ x
Math.sqrt(*x*)	បោះតំលៃ រឹសការ៉េ នៃ x
Math.tan(*x*)	បោះតំលៃ tangent នៃ x

Module 5: Browser Object Model

Browser Object Model គឺជាការសិក្សាពី Object ដែលធ្វើការជាមួយនឹង Browser window។ Browser Object Model ត្រូវបានបង្កើតឡើងដោយបណ្តុំនៃ Objects ដែលជាប់ទាក់ទងគ្នាមួយចំនួន ដូចខាងក្រោម៖



window object

ជា Object ដែលតំនាង Browser window មួយទាំងមូល ប៉ុន្តែមិនគ្រប់គ្រងទៅលើ content ដែលមានក្នុង window នោះទេ។ គេប្រើវាដើម្បីធ្វើការប្តូរទីតាំង ប្តូរទំហំ និងប្រើប្រាស់នូវលក្ខណៈរបស់ Browser។

window object មានផ្ទុកនូវ properties ដូចជា៖

Table: Window Object Properties

Property	ការពិពណ៌នា
closed	true កាលណា window ត្រូវបានបិទ និង false កាលណាមិនទាន់បិទ
defaultStatus	សំរាប់ចាប់យកតំលៃ និងបញ្ចូលតំលៃទៅអោយ status bar របស់ browser ពេលដែលគ្មានការ ប្រើប្រាស់ទៅលើ status
document	បោះនូវ document object
frames[ ]	បោះនូវ frames array object
history	បោះនូវ history object
length	បោះនូវ ចំនួន framesទាំងអស់ក្នុង window
location	បោះនូវ location object

name	ជាឈ្មោះតំនាងអោយ window រឺក៏ frame។ ឈ្មោះត្រូវបានបញ្ជាក់ពេលប្រើ window.open() និងពេលបង្កើត frameដោយប្រើ name attribute។
navigator	បោះនូវ naivagator object
opener	បោះនូវ window object ដែលបានបើក window របស់ខ្លួន
parent	បោះនូវ window object ដែលមាន window រឺ frame របស់ខ្លួន។
screen	បោះនូវ screen object
self	បោះនូវ window object របស់ខ្លួន
status	សំរាប់ចាប់យកតំលៃ និងបញ្ចូលតំលៃទៅអោយ status bar របស់ browser ពេលដែលគ្មានការ ប្រើប្រាស់ទៅលើ status
top	បោះនូវ window obejct ផ្នែកខាងក្រៅ ដែលផ្ទុក window object របស់ខ្លួន។
window	បោះនូវ window object របស់ខ្លួន

Methods in window object

Table: Window Object Methods

Method	ការពិពណ៌នា
`void alert( message)`	បង្ហាញនូវ Message Box
`void blur( )`	ធ្វើអោយបាត់បង់ focus។
`void clearInterval( intervalId)`	សំរាប់លុបចោលនូវ តំណើរការរបស់ method setInterval() ដោយបញ្ជាក់នូវ intervalId.
`void clearTimeout( timeoutId)`	សំរាប់លុបចោលនូវ តំណើរការរបស់ method setTImeout() ដោយបញ្ជាក់នូវ timeoutId.
`void close( )`	សំរាប់បិទផ្ទាំង window។
`boolean confirm( question)`	បង្ហាញនូវផ្ទាំង message ដែលមានសារ question ហើយមានប៊ូតុង ២គឺ OK និង Cancel។ ពេលចុច OK វាបោះតំលៃ true និង ពេលចុច Cancel វាបោះតំលៃ false។
`void focus( )`	សំរាប់ធ្វើ cursor មានលើ browser ហើយក៏ ធ្វើអោយ browser លេចខាងមុខគេផងដែរ។
`void moveBy( dx, dy)`	រំកិលទីតាំងនៃ window ដោយធៀបនឹងទីតាំងដើម ដែលចំងាយគិតជា pixel។
`void moveTo( x, y)`	រំកិលទៅទីតាំងច្បាស់លាស់ x,y
open(*url*, *name*, *features*)	បើកនូវ address តាម url ក្នុង window ដែលបានអោយឈ្មោះ name និងអាចកំនត់លក្ខណៈ អោយ window ថ្មីនោះដោយប្រើ features ដែលត្រូវបានដាក់ក្នុងលក្ខណៈជា string ។
`void print( )`	សំរាប់បង្គាប់អោយ window print នូវ current page។

prompt(*message*, *default*)	បង្ហាញជា InputBox ដែលមានសារជា message ហើយក្នុង Textbox មានតំលៃ default ។ ហើយបោះតំលៃជា string នូវអ្វីដែល user បានបញ្ចូល។
void resizeBy(*dw*, *dh*)	ប្តូរទំហំរបស់ window ដោយអាស្រ័យនឹងទំហំចាស់។
void resizeTo(*width*, *height*)	ប្តូរទំហំរបស់ window ដោយបញ្ជាក់ទំហំទទឹង និងកំពស់ជា pixel។
void scroll(*x*, *y*)	រំកិល scroll របស់ browser window ទៅកាន់ទីតាំង x និង y។
void scrollBy(*dx*, *dy*)	រំកិល scroll របស់ browser window ទៅកាន់ទីតាំងដោយរកិលបន្ថែម dx និង dy។
void scrollTo(*x*, *y*)	រំកិល scroll របស់ browser window ទៅកាន់ទីតាំង x និង y។
setInterval(*code*, *interval*)	បោះនូវ intervalId តំនាងអោយ Interval ដែលបានបង្កើត។ វាធ្វើការ execute នូវកូដ code រាល់ពេល interval ម្តង ហើយម្តងទៀតរហូតដល់ Interval ត្រូវបានលុបចោលដោយប្រើ clearInterval(InvervalID);
setTimeout(*code*, *delay*)	បោះនូវ timeoutId តំនាងអោយ timeout ដែលបានបង្កើត។ វាធ្វើការ execute នូវកូដ code នៅពេលដែល timeout មកដល់។ វាអាចនឹងទប់ កុំអោយតំណើរការបានដោយលុបចោលនូវ timeoutId ដោយប្រើ clearTimeout(timeoutID);

history object

ជា object ដែលធ្វើការលើ history ដែលធ្វើការលើឆ្ពោះទៅកាន់ page ដែលបានរួចម្តងហើយ ។ history មាននូវ methods ដូចជា៖

Table: History Object Methods

Method	ការពិពណ៌នា
back( )	ទៅកាន់ page ពីមុន ដែលបានចូលរួចហើយដែលមានក្នុង history របស់ browser។
forward( )	ទៅកាន់ page បន្ទាប់ ដែលបានចូលរួចហើយដែលមានក្នុង history របស់ browser។
go(*n*)	ទៅកាន់លំដាប់ page ណាមួយបន្ទាប់ពី page ដែលកំពុងមើល។លេខ ១ តំនាងអោយទៅមុខបន្ទាប់ និង -១ សំរាប់ត្រឡប់ទៅក្រោយ។

screen object

ជា object ដែលមាន properties សំរាប់ទាញយកតំលៃដែលទាក់ទង ទៅនឹងពត៌មានរបស់ computer display ដូចជា screen resolution និង color depth ជាដើម។

Screen object មាន properties ដូចជា៖

Table: Screen Object Properties

Property	ការពិពណ៌នា
availHeight	កំពស់នៃ screen ដែលអាចប្រើប្រាស់បាន
availWidth	ទទឹងនៃ screen ដែលអាចប្រើប្រាស់បាន
colorDepth	កំរិតពណ៌នៃ screen ដែលប្រើប្រាស់
height	កំពស់សរុបនៃ screen
width	ទទឹងសរុបនៃ screen

location object

ជា object ដែលតំនាងអោយ URL ដែលបាន load ទៅកាន់ window ហើយវាបានបំលែង URLទៅជាច្រើនបំនែកផ្សេងគ្នា៖

location មាននូវ properties ដូចជា៖

Table: Location Object Properties

Property	ការពិពណ៌នា
hash	វាបោះនូវផ្នែកនៃ URL ដែលមានផ្នុកសញ្ញា # ពីមុខ ឧ.http://www.abc.com/home.html#faq ដូចនេះ hash គឺមានតំលៃស្មើនឹង faq
host	វាបោះឈ្មោះរបស់ server ដូចឧទាហរណ៌ខាងលើ (www.abc.com)
hostname	វាបោះនូវ host name របស់ server (abc.com)
href	សំរាប់កំនត់ និងបោះនូវផ្នែកទាំងមូលរបស់ URL
pathname	បោះនូវផ្នែកពីក្រោយ host ដូចជា http://www.abc.com/images/img1.jpg ក្លាយជា “/images/img1.jpg”
port	បោះនូវ port ដែលប្រើសំរាប់ទំនាក់ទំនងលើ web ដូចជា http://www.abc.com:8080/index.html ស្មើនឹង 8080
protocol	បោះនូវផ្នែកខាងមុខសញ្ញា //។
search	បោះនូវផ្នែកខាងក្រោយនៃសញ្ញា ? ដូចជា http://www.abc.com/index.html?page=1,section=2 វាបោះនូវ “?page=1,section=2”

location មាននូវ methods ដូចជា៖

Table: Location Object Methods

Methods	ការពិពណ៌នា
reload(*force*)	load page ឡើងវិញពី cache កាលណា force ស្មើនឹង false និងពី server កាលណា force ស្មើនឹង true។
replace(*url*)	ប្តូរនូវ url ថ្មីដោយមិនអោយ history តាមដាន លើការប្តូរ page នោះទេ
assign(*url*)	ដូចទៅនឹងការកំនត់តំលៃអោយ href property ដែរគឺវាបើកនូវ url ថ្មី

navigator object

ជា object ដែលមានផ្តុកពីពត៌មានរបស់ Web browser. ដោយសារគ្មាន standard ច្បាស់លាស់សំរាប់គ្រប់គ្រង BOM ទើបធ្វើអោយ navigator object មាន properties និង methods មិនដូចគ្នាពី browser មួយទៅ browser មួយ។

Table: Navigator Object Properties

property/method	ការពិពណ៌នា	Browser	
appCodeName	បោះជា string ដែលតំនាងអោយ code name របស់ browser	IE	Moz
appName	string ដែលតំនាងអោយឈ្មោះផ្លូវការរបស់ browser	√	√
appMinorVersion	string តំនាងអោយពត៌មានបន្ថែម	√	√
appVersion	string តំនាងអោយជំនាន់របស់ browser រឺ OS	√	
browserLanguage	string តំនាងអោយភាសារ របស់ browser	√	√
cookieEnabled	boolean បង្ហាញថាតើ cookies ត្រូវបានបើករឺទេ	√	
cpuClass	string តំនាងអោយ class នៃ CPU	√	√
javaEnabled()	boolean បង្ហាញថាតើ java ត្រូវបានបើករឺទេ	√	
language	string តំនាងអោយជំនាន់របស់ browser		√
mimeTypes	array នៃmimetypes ដែលមានក្នុង browser		√
onLine	boolean បង្ហាញថាតើ browser ត្រូវបានភ្ជាប់ internet រឺទេ	√	
oscpu	string តំនាងអោយOS រឺ CPU		√
platform	string តំនាងអោយ computer platform	√	√
plugins	array នៃ plugins ដែលបានបញ្ចូលក្នុង browser	√	√
preference()	method សំរាប់កំនត់ preferences របស់ browser		√
product	string ដែលតំនាងអោយឈ្មោះរបស់ product		√
productSub	string ដែលតំនាងអោយពត៌មានបន្ថែមលើ product		√
systemLanguage	string ដែលតំនាងភាសារបស់ OS	√	
taintEnabled()	boolean បង្ហាញថាតើ data-tainting ត្រូវបានបើករឺនៅ	√	√
userAgent	string តំនាងអោយ user-agent	√	√
userLanguage	string ដែលតំនាងភាសារបស់ OS	√	

userProfile	object ដែលអាចអោយចូលប្រើប្រាស់ user profile	√	
vendor	ឈ្មោះដែលតំនាងអោយមាកនៃ browser		√
vendorSub	string ដែលតំនាងអោយពត៌មានបន្ថែម		√

document object

Document Object ជា object ដែលផ្ទុកនូវ properties និង methods យ៉ាងសំខាន់ក្នុងការកែប្រែលក្ខណៈរបស់ Page ដូចជា ប្តូរពណ៌របស់ Link, ពណ៌អក្សរ, ពណ៌ផ្ទៃ, ជាពិសេសវាអាចធ្វើការងារជាមួយ cookie បាន។ លើសពីនេះវាអាច access ទៅកាន់ collection ដូចជា form, image បាន។

Table: Document Object Properties

Property	Description
alinkColor	កំនត់នូវ ពណ៌របស់ link ដែលកំពុងតែបើក
anchors[ ]	បោះនូវ array នៃ Anchor object
applets[ ]	បោះនូវ array នៃ Applet object
bgColor	កំនត់នូវពណ៌ background
cookie	A string-valued property with special behavior that allows the cookies associated with this document to be queried and set.
embeds[ ]	បោះនូវ array នៃ Embeds object
fgColor	កំនត់នូវពណ៌អក្សរ
forms[ ]	បោះនូវ array នៃ Form object
images[ ]	បោះនូវ array នៃ image object ដែលមានក្នុង tag <img/>
lastModified	បញ្ជាក់ពីពេលវេលាក្រោយបង្អស់ដែលបានកែ page
linkColor	កំនត់នូវ ពណ៌របស់ link ដែលមិនដែលចុចនៅឡើយ
links[ ]	បោះនូវ array នៃ link object ដែលមានក្នុង tag <a/>
location	បោះនូវទីតាំង URL របស់ file
plugins[ ]	បោះនូវ array នៃ plugins object
title	សំរាប់អាន រឺកំនត់អក្សរក្នុង tag <title>
URL	បោះនូវទីតាំង URL របស់ file
vlinkColor	កំនត់នូវ ពណ៌របស់ link ដែលចុចរួច

Table: Document Object Method

Methods	Description
write(*value, . . .*)	ប្រើសំរាប់បញ្ចូល string មួយ រឺ ច្រើនទៅក្នុង document។
writeln(*value, . . .*)	ប្រើសំរាប់បញ្ចូល string មួយ រឺ ច្រើនទៅក្នុង document ដោយបន្ថែម new line។

Module 6: Document Object Model

Document Object Model គឺជា object មួយដែលគ្រោងនូវ page ទាំងមូលអោយជា document មួយដែលបង្កើត ដោយរចនាសម្ព័ន្ធនៃ node។ វាក៏បានផ្តល់នូវ method និង property សំរាប់ទាញយក នូវ element object មកប្រើ កែប្រែ បន្ថែម node លុប node និងការងារដទៃៗ សំរាប់កាងារលើ form។

Properties and Methods in DOM

Table: Document Object Model

Property/Method	Type Return Type	Description
nodeName	String	ឈ្មោះរបស់ node វាត្រូវបានកំនត់ដោយពឹងផ្អែកលើប្រភេទនៃ node
nodeValue	String	តំលៃនៃ node វាត្រូវបានកំនត់ដោយពឹងផ្អែកលើប្រភេទនៃ node
nodeType	Number	មួយក្នុងចំនោមតំលៃ node type constant
ownerDocument	Document	ចង្អុលទៅកាន់ document ដែលជា parent នៃ node
firstChild	Node	ចង្អុលទៅកាន់ node តំបូងនៅក្នុងបញ្ជី childNodes
lastChild	Node	ចង្អុលទៅកាន់ node ដែលនៅក្រោយគេនៃបញ្ជី childNodes
childNodes	NodeList	បញ្ជីនៃគ្រប់ child node ទាំងអស់
previousSibling	Node	ចង្អុលទៅកាន់ previous sibling វាបោះតំលៃ null កាលណាវាជា sibling តំបូងគេ
nextSibling	Node	ចង្អុលទៅកាន់ next sibling វាបោះតំលៃ null កាលណាវាជា sibling ក្រោយគេ
hasChildNodes()	Boolean	true កាលណា childNodes មាន node មួយ រឺច្រើន
attributes	NamedNodeMap	ផ្ទុកនូវ attr objects ដែលតំនាងអោយ properties នៃ elements មួយ វាប្រើសំរាប់តែ Element nodes ប៉ុណ្ណោះ
appendChild(node)	Node	បន្ថែម Node ទៅកាន់ខាងចុង នៃ childNodes
removeChild(node)	Node	លុប node ពី childNodes
replaceChild(newnode, oldnode)	Node	ជំនួស oldnode ក្នុង childNodes ដោយ newnode
insertBefore(newnode,refnode)	Node	បន្ថែម newnode ពីមុន refnode ក្នុង childNodes

<u>ចំណាំ៖</u>

`NodeList` គឺតំនាងអោយ array នៃ nodes ដែល index ដោយលេខ ប្រើសំរាប់តំនាងអោយ child nodes នៃ elements មួយ។

`NamedNodeMap` គឺតំនាងអោយ array នៃ nodes ដែលអាច index ដោយប្រើបានទាំងលេខ និងដោយឈ្មោះ ប្រើសំរាប់តំនាងអោយ attributes នៃ elements

ចំពោះតំលៃនៃ node type constant នីមួយៗមានដូចជា៖

NodeType	Node Type Description
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

Accessing Element Object

ក្នុងចំនុចនេះយើងនឹងសិក្សាពី ការទាញយក element object មកប្រើដោយ ការប្រើប្រាស់នូវ documentElement, childNodes, firstChild, lastChild, nextSibling, previousSibling,

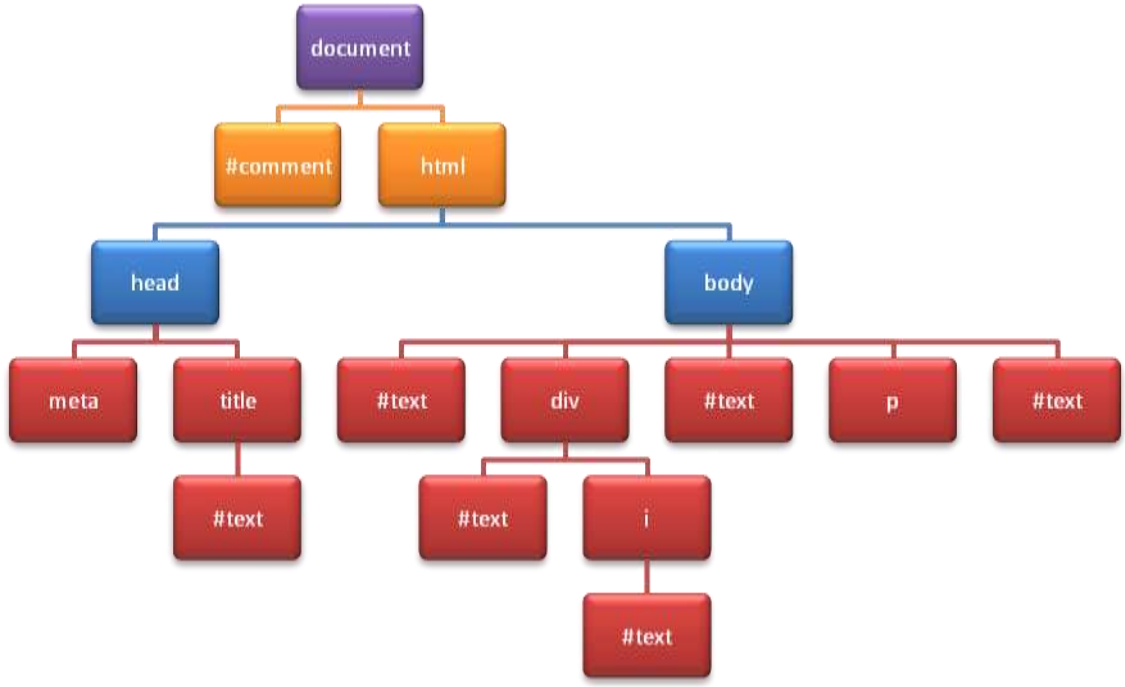
ឧបមាថាយើងមាន នូវកូដ HTML ដូចខាងក្រោម៖

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>DOM Lesson</title>
</head>

<body>
      My First Content
      <div>
            Hello
            <i>
                  World
            </i>
      </div>
      My First Paragraph
      <p>
```

```
            Welcome
      </p>
</body>
</html>
```

កូដខាងលើអាចបំលែងជា រចនាសម្ព័ន្ធនៃ node ដោយប្រើប្រាស់នូវ DOM ដូចខាងក្រោម៖

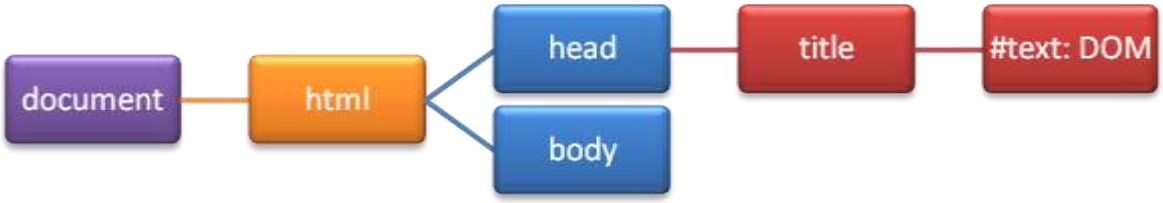


By Hierarchy

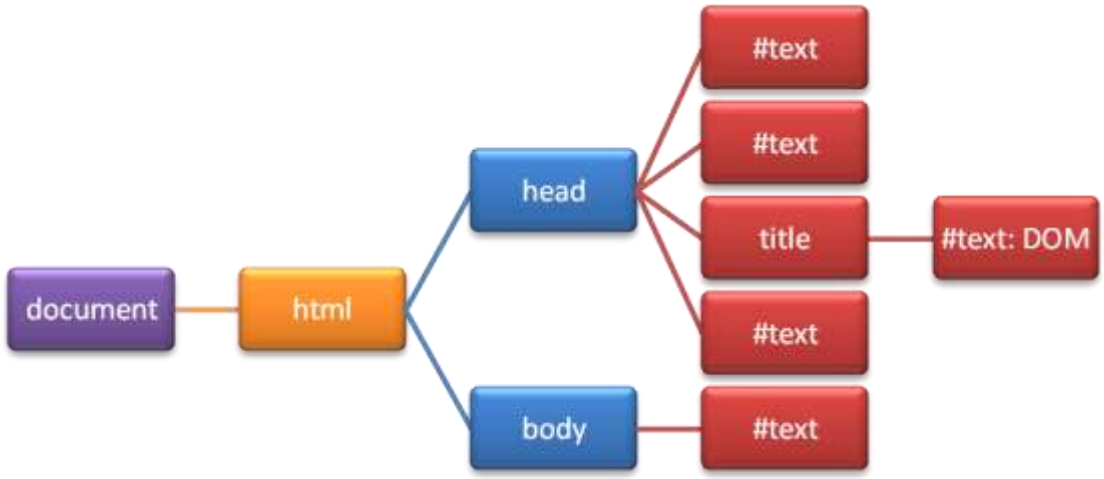
ឧបមាថាយើងមាន HTML ដូចខាងក្រោម៖

```
<html>
      <head>
            <title> DOM </title>
      </head>
      <body>
      </body>
</html>
```

រចនាសម្ព័ន្ធក្នុង Internet Explorer



ដោយសារតែក្នុង Firefox រាល់ចន្លោះរបស់ tag អោយតែមាន white space នឹងបង្កើតអោយមាន #text node ដូច្នេះទើបក្នុង Firefox មានលក្ខណៈដូចខាងក្រោម៖



វិធីក្នុងការទាញយក html element object:

```
var oHTML=document.childNodes[0];
var oHTML=document.documentElement;
var oHTML=document.firstChild;
var oHTML=document.lastChild;
```

ដើម្បីទាញយកនូវ head element យើងអាចប្រើបានដោយរបៀបដូចខាងក្រោម៖

```
1:    var oBody=oHTML.firstChild;
2:    var oBody=oHTML.childNodes[0];
3:    var oBody=oHTML.childNodes.item(0);
```

ដើម្បីទាញយកនូវ body element យើងអាចប្រើបានដោយរបៀបដូចខាងក្រោម៖

```
1:    var oBody=document.body;
2:    var oBody=oHTML.lastChild;
3:    var oBody=oHTML.childNodes[0];
4:    var oBody=oHTML.childNodes.item(1);
```

Access Element By Element's Id

ជាការទាញយក element object ណាមួយដោយការប្រើនូវ id attribute។ id attribute មួយដែលមិនអាចមានតំលៃដូចគ្នាបានទេ។ ក្នុងករណីដែលមានការមានតំលៃដូចគ្នា element ដែលនៅមុនគេ ជាអ្នកមានអាទិភាពជាង។

Syntax:

```
var oElement=document.getElementById("elementID")
```

ឧបមាថាយើងមាន HTML ដូចខាងក្រោម៖

```
<html>
```

```
            <head>
                  <title> DOM </title>
            </head>
            <body>
                  <h1 id=”headerText”>Web Content </h1>
                  <p id=”content”>Welcome To My Site </p>
            </body>
      </html>
```

ដើម្បីទាញយក p element object មកប្រើយើគ្រាន់តែបញ្ជាក់ id របស់ element នោះជាការស្រេច

```
      var oPContent=document.getElementById(“content”);
```

ចំពោះ h1 element object៖

```
var oH1HeaderText=document.getElementById(“headerText”);
```

Access Element By Element’s Name

គេប្រើវាសំរាប់ `element` ដែលតម្រូវអោយមានឈ្មោះដូចគ្នាដូចជា `radio, checkbox`
Syntax:

```
      var oElement=document.getElementsByName(“elementName”)
```

oElement ផ្ទុកតំលៃជា array ដែលផ្ទុកនូវ element នីមួយៗ។

Radio Implementation

```
<form>
      <input type="radio" name="radAge" value="14" checked="checked"
/>14<br />
      <input type="radio" name="radAge" value="15" />15<br />
      <input type="radio" name="radAge" value="16" />16<br />
      <input type="radio" name="radAge" value="17" />17<br />
      <input type="button" onclick="showValue()" value="ShowAge" />
</form>
<script>
function showValue(){
      var oAge=document.getElementsByName("radAge");
      for(i=0;i<oAge.length;i++){
            if(oAge.item(i).checked==true){
                  alert(oAge.item(i).value);
            }
      }
}
</script>
```

Checkbox Implementation

```
<form>
      <input type="checkbox" value="1" name="chkNumber" />1<br />
      <input type="checkbox" value="2" name="chkNumber" />2<br />
      <input type="checkbox" value="3" name="chkNumber" />3<br />
      <input type="button" value="Get Number" onclick="getNumber()"
/>
</form>
<script type="text/javascript">
      var aNumber=new Array();
      var oChkNumber=document.getElementsByName("chkNumber");
      function getNumber(){
            for(i=0;i<oChkNumber.length;i++){
                  if(oChkNumber.item(i).checked==true) {
                        aNumber.push(oChkNumber.item(i).value);
                  }
            }
            alert(aNumber);
      }
</script>
```

Access Element By Element's TagName

ជា method សំរាប់ ទាញយកនូវ array គ្រប់ element ដែលមាន tag name ដែលបានបញ្ជាក់៖

Syntax

```
array oElementObject.getElementByTagName("tagName");
```

ឧទាហរណ៍៖

យើងចង់ទាញយកគ្រប់ element object ណាដែលមាន tag ជា p ដែលបិតក្នុង tag មួយដែលមាន Id ស្មើនឹង content មកដាក់ក្នុង array មួយឈ្មោះថា aPContents

```
...
<div id="content">
      <h2> First Article </h2>
      <p>First Text</p>
      <h2> Second Article </h2>
      <p>Third Text</p>
      <h2> Third Article </h2>
      <p>Third Text</p>
</div>
...
var oContent = document.getElementById("content");
var aPContents=oContent.getElementsByTagName("p");
```

ពេលនេះយើងទទួលបាននូវ array មួយដែលមានផ្ទុកវាល់ element object ដែលមាន tag ជា p យើងអាចរាប់ចំនួនធាតុក្នុង array នោះបាន

```
alert(aPContents.length);// display 3
```

យើងអាចចាប់យក element object ណាមួយមកប្រើដោយបញ្ជាក់នូវ index របស់វា។ ឧបមាយើងនឹងយក p tag ដែលមានទីតាំងតំបូងគេ

```
var oFirstP=aPContents[0];
```

Using document.all

document.all ជា property មួយដែលបោះនូវ array នៃ node ទាំងអស់ក្នុង document object តែមិនគិត #text node នោះទេ។ property នេះ support តែក្នុង Internet Explorer ប៉ុណ្ណោះ។

ឧបមាថាយើងមានកូដ HTML ដូចនឹង HTML កូដក្នុងដើមមេរៀនដែរ ហើយយើងសរសេរកូដ javascript ដូចខាងក្រោមនេះ៖

```
var oAll=document.all;
for(i=0;i<oAll.length;i++){
      document.writeln(oAll.item(i).nodeName);
}
```

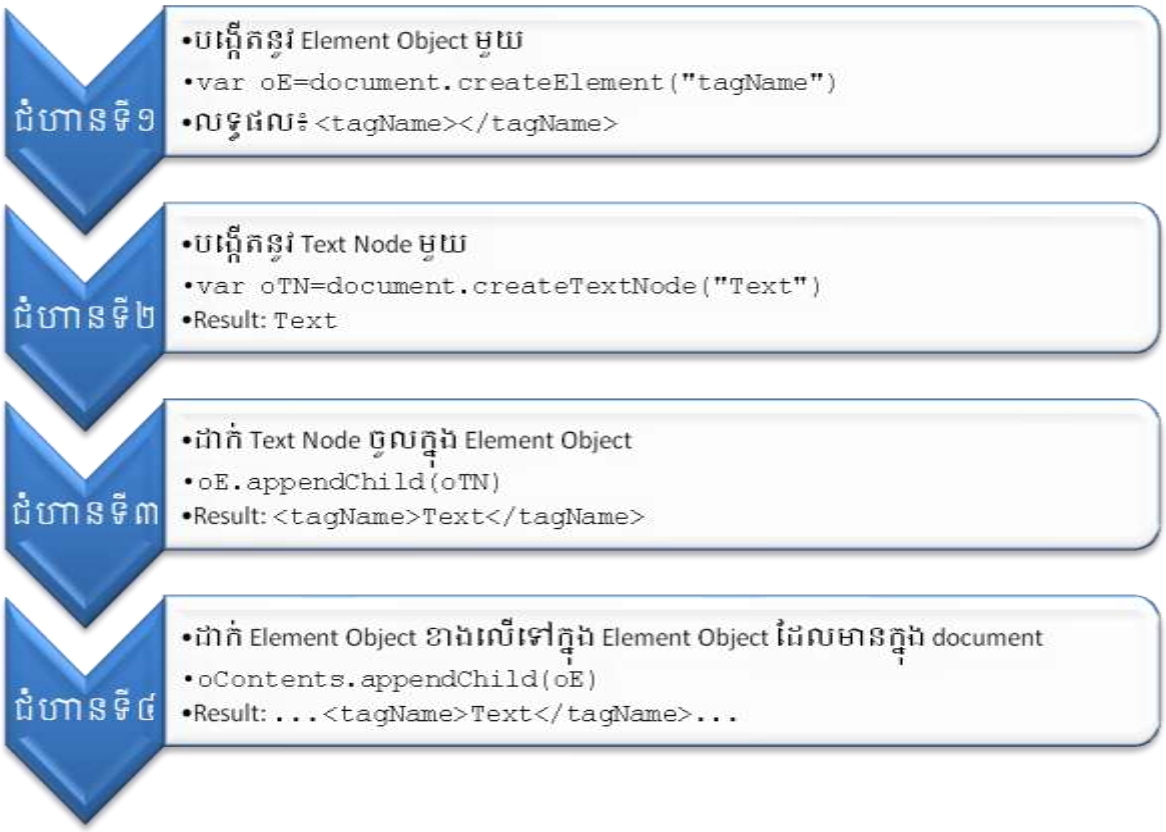
វានឹងចេញដូចខាងក្រោម៖

```
#comment
HTML
HEAD
TITLE
META
BODY
DIV
I
P
PRE
SCRIPT
```

Creating andm Manipulate nodes

Creating node

ជំហានក្នុងការបង្កើត និងរៀបចំ node មួយ



Syntax

ឧទាហរណ៍៖

ឧបមាថាយើងមានកូដ HTML ដូចខាងក្រោម៖

```
<html>
      <head>
            <title>Create a Complete Node</title>
      </head>
      <body>

      </body>
</html>
```

យើងចង់ដាក់នូវដាក់នូវ `<h1>Welcome</h1>` ទៅក្នុង bodyយើងត្រូវអនុវត្តដូចខាងក្រោម៖

```
1     var oH1=document.createElement("h1");
2     var oTextNode=document.createTextNode("Welcome");
3     oH1.appendChild(oTextNode);
4     document.body.appendChild(oH1);
```

Using removeChild(), replaceChild(), insertBefore()

Syntax

```
oElementObject.removeChild(oChildElementObject)
oElementObject.replaceChild(oNewChild,oOldChild)
oElementObject.insertBefor(oNewChild,oReferenceChild)
```

ឧបមាយើងមាន HTML Code ដូចខាងក្រោម៖

```
...
<body>
      <ul id=”list”>
            <li>Item 0</li>
            <li>Item 1</li>
            <li>Item 2</li>
            <li>Item 3</li>
      </ul>
</body>
...
```

មុននឹងធ្វើការលើការលុប ការជំនួស យើងគួរបង្កើតនូវ variable ពីរសិនគឺ ១ សំរាប់ ul object និង ទី២ គឺសំរាប់ផ្ទុក array នៃធាតុ li ក្នុង ul ៖

```
var oList = document.getElementById(“list”);
var oListItems = oList.getElementsByTagName(“li”);
```

ពេលនេះយើងមាន oList ដែល reference ទៅកាន់ ul និង oListItems ផ្ទុករាល់ធាតុរបស់ ul

+ វិធីលុប៖

```
      oList.removeChild(oListItems[2]);
```

+ វិធីប្តូរ

```
      var oNewItem=document.createElement(“li”);
      var oTextNode=document.createTextNode(“New Item”);
      oNewItem.appendChild(oTextNode);

      oList.replaceNode(oNewItem,oListItems[2]);
```

+ វិធីបន្ថែមធាតុមុន reference element

```
      var oNewItem=document.createElement(“li”);
      var oTextNode=document.createTextNode(“New Item”);
      oNewItem.appendChild(oTextNode);


      oList.insertBefore(oNewItem,oListItems[2]);
```

Using createDocumentFragement()

គឺជាការបង្កើតនូវ កន្លែងផ្ទុកបណ្តោះអាសន្ន សំរាប់រាល់ Node ដែលយើងបំរុងនឹង ដាក់ចូលក្នុង node ដាក់មួយទៀត។ គេប្រើវាក្នុងពេលដែលមានការបន្ថែមនូវ node ច្រើន បើយើងមិនប្រើវា វាអាចបណ្តាលអោយចំនួននៃការ refresh របស់ screen ក៏ច្រើនដែលនាំអោយមានភាពយឺត។ ពេលយើងផ្ទុកវាក្នុង documentFragement វានឹងធ្វើអោយចំនួន នៃការ refresh ម្តងប៉ុណ្ណោះ។

Syntax:

```
var oFragement= document.createDocumentFragement();
```

ឧទាហរណ៍៖

```
var oFragment=document.createDocumentFragment();
for(i=0;i<15;i++){
      var oNewNode=document.createElement("h1");
      var oTextNode=document.createTextNode("Head " + i);
      oNewNode.appendChild(oTextNode);
      oFragment.appendChild(oNewNode);
}
document.body.appendChild(oFragment);
```

Module 7: Error Handling

Kind of Error

ជាធម្មតា error មានពីរប្រភេទគឺ៖ syntax errors និង runtime errors។

Syntax errors ជាកំហុសដែលកើតមានឡើងពេល interpret។ វាកើតឡើងដោយសារ unexpected character ក្នុង code ដូច្នេះវាធ្វើអោយការ compile រឺ interpret មិនបានសំរេចនោះទេ។

```
example : alert(“Welcome”;
```

កាលណាមានកំហុសនេះកើតឡើង កូដដែលមានប្លុក script រឺ function នោះទាំងអស់មិនតំណើរការទេ។ តែចំពោះ ប្លុក script រឺ function ដទៃ និង external javascript មិនមានប៉ះពាល់នោះទេ។

Runtime errors រឺ exception ជាកំហុសដែលកើតឡើងពេលកំឡុងពេលតំណើរការ គឺក្រោយពេល compile រឺ Interpret។ វាកើតឡើងដោយសារការ ប្រើកូដមិនត្រឹមត្រូវ ដូចជាការហៅ method រឺ function ដែលមិនមាន, ការ reference ទៅកាន់ object ដែលមិនមាន, ការទាញយក Property ដែលមិនមានក្នុង object ណាមួយ។

```
example : window.showMyName();
```

វាប៉ះពាល់ដល់ ប្លុក javascript រឺ function ដែលវាស្ថិតនៅប៉ុណ្ណោះ ហើយប៉ះពាល់ដល់រាល់កូដណាដែលនៅខាងក្រោយវាប៉ុណ្ណោះ។

Handling Errors

ជាវិធីសាស្ត្រ ក្នុងការគ្រប់គ្រងកំហុសដែលកើតមាន។ ក្នុង JavaScript មានវិធីសាស្ត្រពីរយ៉ាង ក្នុងការគ្រប់គ្រងកំហុសទាំងនោះគឺ៖ onerror event របស់ BOM សំរាប់ window object។ វិធីម្យ៉ាងទៀតគឺ `try...catch` statement ដើម្បីដោះស្រាយជាមួយនឹងកំហុសប្រភេទ runtime error រឺ exception។

onerror event handler

event error ជា event របស់ window object ដែលកើតឡើងពេលដែលមានកំហុសកើតឡើងនៅលើ page។

ឧទាហរណ៍៖

```
<head>
      <script type=”text/javascript”>
            window.onerror=function(){
                  alert("got error");
```

```
            return true;
      }
      incorrect;
</script>


<body onload=”incosistentMethod()”>
</body>
```

`return true` ត្រូវបានប្រើសំរាប់ការពារកុំអោយ browser ចេញផ្ទាំង error។
យើងចាប់យកពតិមានលំអិតបន្ថែមតាមរយៈ ការប្រើប្រាស់ argument ៣
ដែលវាបោះដោយស្វ័យប្រវត្តិ មានដូចជា៖

- Error message ដូចទៅនឹងសារដែលបង្ហាញនៅ browser
- URL ឈ្មោះ និងទីតាំង file ដែលមានកំហុស
- Line number លេខបន្ទាត់របស់កូដដែលមានកំហុស

ឧទាហរណ៍៖

```
<head>
      <script type=”text/javascript”>
            window.onerror=function(sMessage, sUrl, sLine){
                  alert("An error occured:\n” + sMessage + “\nURL: ” +
sUrl + “\nLine: “ + sLine);
            return true;
      }
      incorrect;
</script>

<body onload=”incosistentMethod()”>
</body>
```

try…catch statement

ជា statement ដែលប្រើសំរាប់គ្រប់គ្រងលើកំហុសដែលកើតឡើងពេល runtime ប៉ុណ្ណោះ
បើមាន

Syntax:

```
      try{
            statements;
      } catch(exception){
            statements_when_error_occur;
      } [finally]{
            statements_always_run.
      }
```

ក្នុងកូដខាងលើ ពេលកំពុងតំណើរការ `try..caught statement` វានឹងតំណើរការកូដ
ដែលមានក្នុង try មុនគេបង្អស់។ កាលណាក្នុងប្លុកនេះមានកំហុសកើតឡើង

វានឹងបញ្ឈប់តំណើរការហើយ លោតចូលក្នុង ប្លុក `catch` បើគ្មានកំហុស វានឹងលោតរំលង `catch`។

`finally` ជាប្លុកអាចមាន និងអាចមិនមាន វាតែងតែតំណើរការជានិច្ច បើទោះជាមាន រឺគ្មាន កំហុសក្នុង `try` ក៏ដោយ។

ចំពោះ `exception` ជា variable ដែលចាំទទួល error object ដែលបោះមកពេលមាន កំហុសកើតឡើង។

Error Object មាន property ចំនួនពីរគឺ៖

- `name` ជា string បោះនូវប្រភេទរបស់កំហុសដែលកើតឡើង
- `message` ជា string បោះនូវសារកំហុស

ឧទាហរណ៍៖

```
try{

}catch(oException){

}
```

Module 8: Form Validation

មធ្យោបាយដែលប្រើដើម្បីនាំយកពត៌មានពី អ្នកប្រើប្រាស់ទៅកាន់ server គឺជា form element។ វាបានផ្តល់នូវទំរង់ web form ដោយការប្រើប្រាស់ `<form/>`, `<input/>`, `<textarea/>`, `<select/>`។ Browser ជាអ្នកបំលែងពី element ទាំងនោះទៅជា textbox, combo box, listbox។

Form Basics

ក្នុង form មួយត្រូវបានផ្សំដោយ elements មួយចំនួនដូចជា៖

1. `<input/>` បានផ្តល់នូវ control object យ៉ាងច្រើនតាមរយៈ type attribute របស់វា ដូចជា`text`, `radio`, `checkbox`, `file`, `password`, `button`, `submit`, `reset`, `hidden`, និង `image`។
2. `<select/>` សំរាប់បង្កើតជា combo box រឺ list box។
3. `<textarea/>` បង្កើតជា textbox ច្រើនបន្ទាត់ដោយកំនត់តាម `cols`, `rows` attribute របស់វា។

Referencing to Form

ដើម្បីទទួលបាន reference ទៅកាន់ form គេមានវិធីដូចខាងក្រោម៖

១ ការប្រើប្រាស់ `document.getElementById()`

```
var oForm = document.getElementById("frmRegistration");
```

២ ការប្រើប្រាស់ form collection នៃ document តាមរយៈការប្រើ លេខលំដាប់ និង ឈ្មោះរបស់ form ដែលកំនត់ដោយ `name` attribute៖

```
var oForm = document.form1;
var oForm = document.forms[0];
var oForm = document.forms["form1"];
```

Referencing to Form Field

ដើម្បីទទួលបាន reference ទៅកាន់ element របស់ form គេមានវិធីដូចខាងក្រោម៖

១ ការប្រើប្រាស់ `document.getElementById()`

```
var oTextName = document.getElementById("txtName");
```

២ ការប្រើប្រាស់ element collection នៃ form object ដែលយើងបាន reference ខាងលើ តាមរយៈការប្រើ លេខលំដាប់ និង ឈ្មោះរបស់ element ដែលកំនត់ដោយ `name` attribute៖

```
var oTextName = oForm.elements[0];
var oTextName = oForm.elements["txtName"];
```

៣ ការយកឈ្មោះរបស់ element ដែលកំនត់ដោយ `name` attribute មកធ្វើជា property របស់ form object៖

```
var oTextName = oForm.txtName
```

ចំពោះឈ្មោះដែល ដកឃ្លាត្រូវប្រើដូចខាងក្រោម៖

```
var oTextName = oForm["txtName"];
```

Form Vaidation

Submitting Forms

ធម្មតាយើងអាច submit form បានតាមរយៈ <html> គេត្រូវប្រើនូវ button ដូចខាងក្រោម៖

```
<input type="submit" value="Submit" />
```

យើងពិនិត្យមើលការ Submit បានតាមរយៈ:

```
<form method="post" action="javascript:alert('Submitted')">
```

យើងប្រើនូវ method submit() របស់ form object ដើម្បីធ្វើការ submit បាន

```
oForm.submit();
```

Submitting Once

ដើម្បីការពារកុំអោយ form មួយធ្វើការ submit លើសពីម្តង ដែលធ្វើអោយចរាចរណ៍ និងការផ្ទៀងផ្ទាត់មានបញ្ហា យើងត្រូវសរសេរកូដខាងក្រោម៖

```
<input type="button" value="Submit"
onclick="this.disabled=true; this.form.submit()"/>
```

Select Element

List Boxes and Combo Boxes

ដើម្បីបង្កើតនូវ List Box និង Combo Box គេត្រូវប្រើនូវ <select/> tags ហើយធាតុនីមួយៗរបស់វាគឺត្រូវប្រើនូវ <option/> tag។

ការបង្កើតនូវ Combo Box:

```
<select name="selAge" id="selAge">
```

```
      <option value="1">18-21</option>
      <option value="2">22-25</option>
      <option value="3">26-29</option>
      <option value="4">30-35</option>
      <option value="5">Over 35</option>
</select>
```

ការបង្កើតនូវ List Box:

```
<select name="selAge" id="selAge" size=”3”>
      <option value="1">18-21</option>
      <option value="2">22-25</option>
      <option value="3">26-29</option>
      <option value="4">30-35</option>
      <option value="5">Over 35</option>
</select>
```

ដើម្បីទាញវាយកប្រើក្នុង JavaScript គេមានវិធីដូចខាងក្រោម៖

```
      oListBox = document.getElementById(“selAge”);
      oListBox = document.forms[“form1”].selAge;
      oListBox = document.forms[0].selAge;
```

Accessing Options Collection

ក្នុង Select Element Object មាននូវ Property មួយឈ្មោះថា options ដែលផ្តុំកនូវ collections របស់ option elements។

```
      var oListBox = document.getElementById(“lstText”);
      alert(oListBox.options[1].value);
```

Getting Text from Option

Option element មាននូវ property មួយសំរាប់យក text ដោយការប្រើនូវ property `text`។

```
      oListBox.options[1].text;
```

Accessing Selected Option

វាក៏មាននូវ property មួយឈ្មោះថា selectedIndex ដែលបោះនូវ index របស់ option ដែលបាន select បើគ្មាន option ណាមួយបាន select ទេ វានឹងបោះនូវតំលៃ -1។

```
      alert(oListBox.options[oListBox.selectedIndex].text);
```

Accessing Multiple Selected Option

យើងក៏អាចប្រើនូវ attribute របស់ option ដើម្បីធ្វើការ ទាញយកនូវ Index របស់ options ដែលបាន select ច្រើន។

```
      var arrSelectedIndex = [];
      for ( var i=0; i<oListBox.options.length; i++){
            if( oListBox.options[i].selected){
                  arrSelectedIndex.push(oListBox.options[i].value;
            }
      }
```

Module 10: Events

Event Handlers/Listeners

Event គឺជាសកម្មភាពជាក់លាក់ណាមួយដែល បណ្តាលមកពី user រឺ browser ផ្ទាល់។ Events មានដូចជា click, load និង mouseover។

Function ដែលគេប្រើសំរាប់ឆ្លើយតបនឹង event មួយត្រូវបានគេហៅថា *event handler* រឺ *event listener*។ Function ដែលឆ្លើយតបនឹង event click ត្រូវបានគេហៅថា onclick event handler។

គេអាចផ្តល់ event handler បានតាមវិធី ២ យ៉ាង គឺតាម JavaScript និង HTML.

ឧទាហរណ៍៖

```
<div id=”oButton” onclick=”eventHandler()>Click Me!</div>
```

រឺ

```
var oButton=document.getElementById(“oButton”);
oButton.onclick=function(){
      alert(“Clicked!”);
}
```

attachEvent(), detachEvent()

method ទាំងពីរនេះ ប្រើបានតែក្នុង internet Explorer ប៉ុណ្ណោះ។ attachEvent() ប្រើសំរាប់ភ្ជាប់ event handler មួយទៅ event មួយ។ ចំនែក detachEvent() ប្រើសំរាប់ផ្តាច់ event handler ពី event មួយវិញ។

Syntax:

```
[Object].attachEvent(“oneventname”, fnHandler)
[Object].detachEvent(“oneventname”, fnHandler)
```

Example

```
var oButton=document.getElementById("oButton1");

var fnClicked=function(){

      alert("OK!");

      oButton.detachEvent(“onclick”,fnClicked);

}

oButton.attachEvent("onclick",fnClicked);
```

យើងអាចភ្ជាប់ event handler ច្រើនទៅកាន់ event មួយតាមរយៈការ attactEvent()បន្ថែមទៀត។

addEventListener(), removeEventListener()

គេប្រើវាក្នុង DOM Compliant Browser ដូចជា Opera, Mozilla Firefox, Netscape តែវាមិនដើរជាមួយនឹង Internet Explorer ទេ។ វាមានតួនាទីដូចនឹង attachEvent() និង detachEvent()ដែរ។

Syntax:

```
[Object].addEventListener("eventname",fnHandler,bCapture)
[Object].removeEventListener("eventname",fnHandler,bCapture)
```

ចំពោះ bCapture សំដៅថាតើ browser ប្រើ bubble event រឺក៏ capture event។ ចំពោះ version ក្រោយៗ ភាគច្រើនប្រើនូវ bubble event ដូច្នេះបើយើងដាក់ true វាមិន Listen ទេ។

Example

```
var oButton=document.getElementById("oButton1");

var fnClicked=function(){

      alert("OK!");

      oButton.detachEvent("click",fnClicked);

}

oButton.attachEvent("click",fnClicked,false);
```

***ចំណាំចំពោះ event onload របស់ body ក្នុង Firefox មិនអាចធ្វើតាមវិធីទាំងប៉ុន្មានខាងលើបានទេ។ គេត្រូវ បង្កើត function មួយឈ្មោះថា onload តែម្តងដូចខាងក្រោម៖

```
function onload(){
      statements;
}
```

Module 11: File System Object

ជាប្រភេទ ActiveX Object មួយដែលប្រើសំរាប់ធ្វើការជាមួយ File System។ វាមានលទ្ធភាពក្នុងការបង្កើត Folder, Text File, Copy Folder, Delete Folder, Delete File ហើយអាចទាញយកពត៌មានពី File, Folder, និង Drive បានទៀតផង។

Working with Text File

រាល់ការងារដែលទាក់ទងនឹង File System Object យើងត្រូវតែ បង្កើតនូវ variable ធ្វើជា instance របស់វា។

Syntax:

```
var oFSO = new ActiveXObject("Scripting.FileSystemObject");
```

Create Text file

ដើម្បីបង្កើតនូវ Text File យើងត្រូវប្រើនូវ method មួយរបស់ fso គឺ createTextFile()។ យើងក៏ត្រូវបង្កើតនូវ variable មួយទៀតផងដែរសំរាប់ទទួលនូវ file object ដែលបាន បោះចេញពី method ខាងលើ។

file object មាននូវ method សំរាប់ការ អានទិន្នន័យ ការបញ្ចូលទិន្នន័យជាដើម។

Syntax ទូទៅ៖

file object មាន method ដូចខាងក្រោម

```
var oTextFile = oFSO.createTextFile("Path",true,true);
```

Write Data Into Text File

យើងត្រូវប្រើ method មួយរបស់ File Object ឈ្មោះថា write() រឺ writeLine()

```
oTextFile.write("Hello My First File.");
oTextFile.close();
```

Open Text File and Read Out

យើងត្រូវប្រើនូវ method openTextFile()ដោយបញ្ជាក់លេខ mode ស្មើនឹង១ មានន័យថាសំរាប់តែអានប៉ុណ្ណោះ(read only) ហើយប្រើនូវ readLine()សំរាប់ការអានម្តងមួយបន្ទាត់ រឺ readAll() សំរាប់ការអានទាំងអស់។ រាល់ការអាន វាបោះតំលៃជា string។

```
var oFSO = new ActiveXObject("Scripting.FileSystemObject");
var oFile = oFSO.openTextFile("Path",1);
alert(oFile.readAll);
```

Appending Text File

យើងត្រូវប្រើនូវ method `openTextFile()` ដោយបញ្ជាក់លេខ mode របស់វា ស្មើនឹង 8 វិញ ហើយប្រើនូវ write និង writeline ដើម្បីបញ្ចូលបន្ត។

Excercises

Exercise I

គោលបំណង

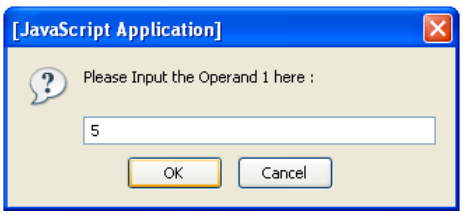
១ អោយចេះប្រើ External JavaScript

២ អោយចេះប្រកាស បញ្ចូល និងទាញយកទិន្នន័យពី variable

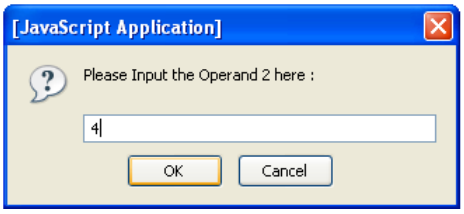
៣ អោយចេះប្រើ InputBox, និងការសរសេរកូដ HTML ដោយប្រើ document.write()

ប្រធាន

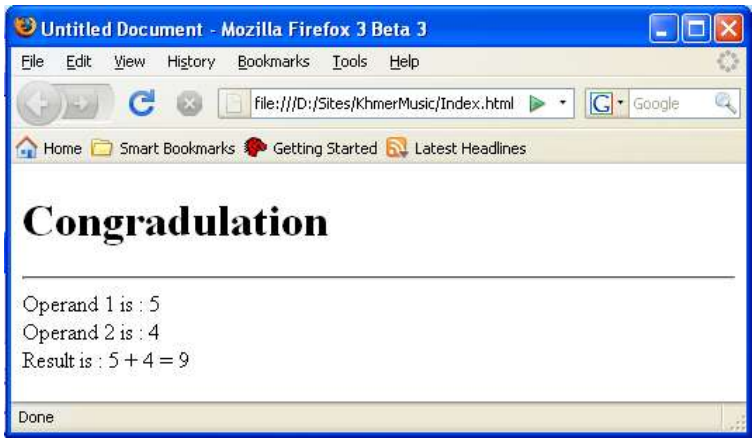
ពេលតំណើរតំបូង វាបង្ហាញនូវ InputBox ដើម្បីអោយវាយបញ្ចូលនូវតួទ១(Operand1)



ពេលចុចលើប៊ូតុង OK វានឹងបង្ហាញនូវ InputBox មួយទៀតសំរាប់វាយបញ្ចូនូវតួទី២(Operand2)



ពេលចុចលើប៊ូតុង OK វានឹងធ្វើការគណនាផលបូករវាងតួទាំងពីរ និងបង្ហាញនូវលទ្ធផលដូចខាងក្រោម៖



លក្ខខ័ណ្ឌ

- ត្រូវប្រើនូវ External JavaScript ហើយ link មកកាន់ HTML Page
- គ្រប់ HTML Code ត្រូវប្រើ document.write() method
- Variable ត្រូវមាន៣

គន្លឹះ

- ប្រើនូវ eval() method ដើម្បីបំលែងពី String ទៅជា Number ដើម្បីធ្វើការគណនា។
- មិនគួរសរសេរ HTML Code ទាំងអស់ក្នុង document.write()តែមួយទេ គួរប្រើមួយបន្ទាត់មួយ។

Exercise 2

- បង្កើតនូវ InputBox ចំនួន3 ដែលសំរាប់ដាក់ពិន្ទុសំរាប់មុខវិជ្ជា Math, Physic, Khmer
- ពិន្ទុទាំងអស់ត្រូវទុកជា Number
- ត្រូវរកមធ្យមភាគនៃពិន្ទុទាំងបី
- ត្រូវបង្ហាញលទ្ធផលដូចខាងក្រោម៖

0	➔	50	: Fail.
50	➔	100	: Passed.

- បង្ហាញនិទ្ទេសដូចខាងក្រោម៖

90	➔	100	: A
80	➔	90	: B
70	➔	80	: C
60	➔	70	: D
50	➔	60	: E
0	➔	50	: F

- និងបង្ហាញនូវលទ្ធផលដូចខាងក្រោម៖

```
Your Result

_____________
Math : 60.5
Physic : 80.3
Khmer : 30.8
Average : 57.2
Result : Passed
Grade : E
```

Exercise 3

- បង្កើតនូវ InputBox ចំនួន 3 សំរាប់ដាក់ Operand1, Operand2, និង Operator
- Operand1, Operand2 ផ្ទុកជា Number និង Operator ផ្ទុកជា String
- Operator អាចផ្ទុកតំលៃដូចជា +,-,*,/,%
- ត្រូវគណានានូវ Operand1 និង Operand2 ទៅតាម Operator
- រួចបង្ហាញនូវលទ្ធផលដូចខាងក្រោម៖

```
Calculation

_________________
Operand1: 6
Operand2: 8
Operator: *
Result: 6 * 8 = 48
```

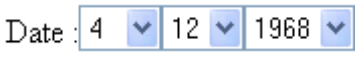
បើ Operator ខុសពីតំលៃដែលត្រូវមាន ត្រូវបង្ហាញដូចខាងក្រោម៖

```
Calculation

_________________
Operand1: 6
Operand2: 8
Operator: efd
Result: 6 efd 8 = Invalid Operator
```

Exercise 4

ចូរអ្នកបង្កើតនូវ Combo Box ចំនួនបីដែលទីមានតំលៃពី 1-31 ទី២មានតំលៃពី 1-12 និង ទី៣ មានតំលៃពី 1960-2009



Exercise 5

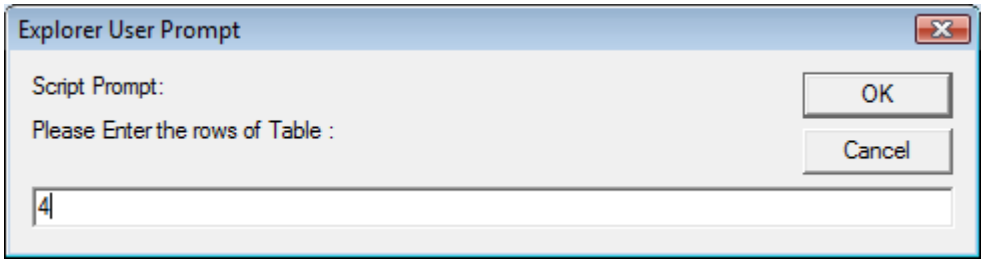
ចូរអ្នកសរសេរកូដដែលតម្រូវអោយ user ត្រូវតែវាយបញ្ចូលនូវ លេខជាដាច់ខាត បើមិនដូច្នោះទេ តម្រូវអោយចេញនូវផ្ទាំង inputbox ម្តងហើយម្តងទៀត រហូតដល់ user វាយលេខ ទើបអាចអោយបន្តបាន។

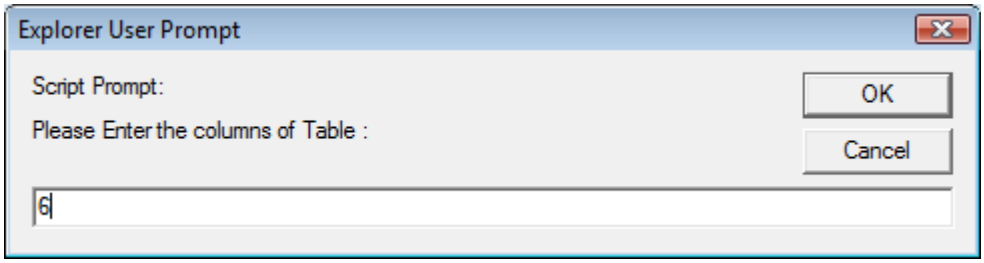
គន្លឹះ គួរប្រើនូវ do while ដើម្បីសំរួលកូដ

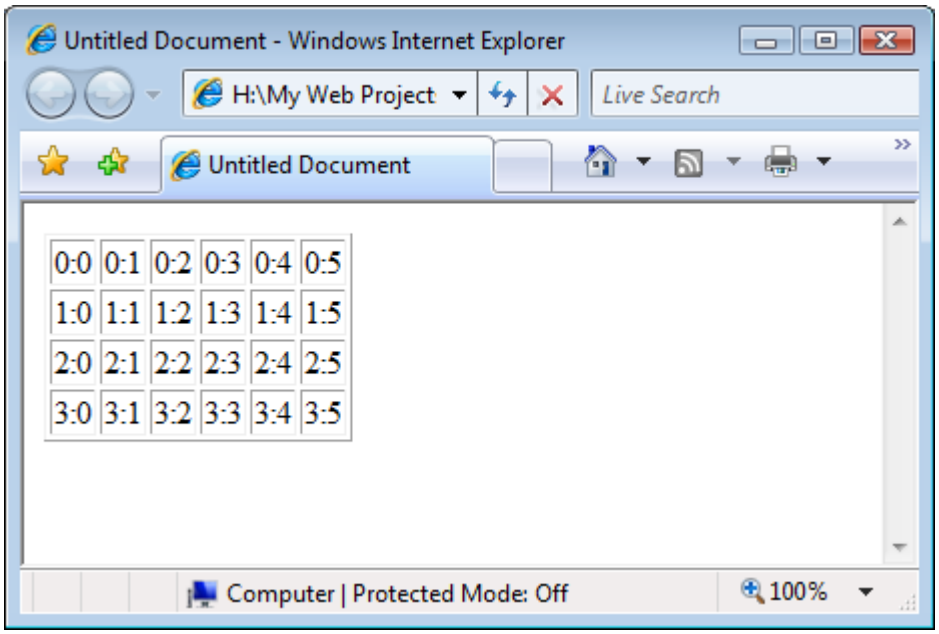
Exercise 6

- ចូរអ្នកបង្កើតនូវ Input Box ចំនួនពីរដែលតម្រូវអោយវាយចំនួន rows និងចំនួន columns
- ប្រើចំនួន rows និង columns ដើម្បីបង្កើតជា Table មួយ។

- ក្នុង Cell នីមួយៗត្រូវមានតំលៃ







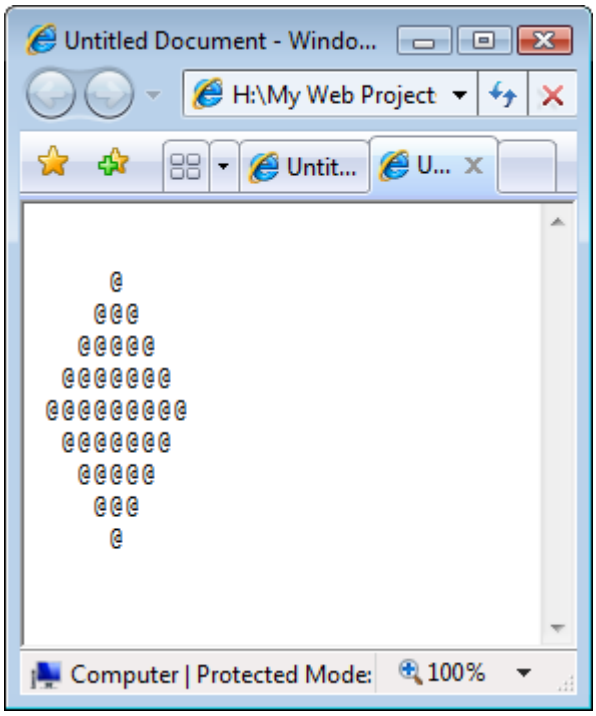
Exercise 7

- ចូរអ្នកបង្កើតនូវ Input Box មួយដែលតម្រូវអោយវាយចំនួន lines ដែលត្រូវបង្ហាញ។
- ចូរអ្នកយកចំនួន rows មកបង្កើតនូវ HTML ដូចខាងក្រោម៖

ឧទាហរណ៍បើ line = 5

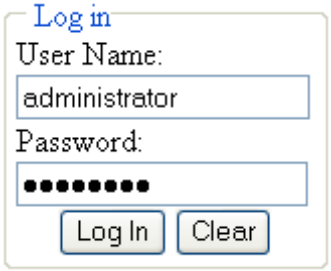
```
1.@          2.@@@@@     3.@          4.   @
  @@           @@@@        @@             @@@
  @@@          @@@         @@@          @@@@@
  @@@@         @@          @@             @@@
  @@@@@        @           @              @
```

ឧទាហរណ៍ line=10



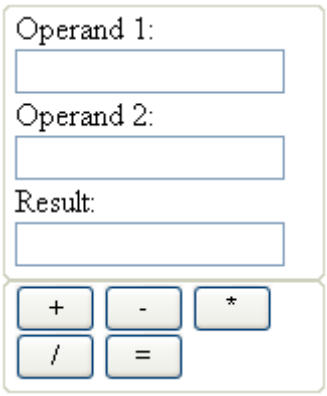
Exercise 8

- ចូរអ្នកបង្កើត TextBox ចំនួន ២
- TextBox ទី១ សំរាប់ដាក់ ឈ្មោះ user
- TextBox ទី២ សំរាប់ដាក់ password
- ចូរអ្នកបង្កើតនូវប៊ូតុង ចំនួន ២
- button ទី១ ឈ្មោះថា Log in
- button ទី២ ឈ្មោះថា Clear
- ពេល user ចុចលើ Button log inបើ Textbox ទាំងពីរ ណាមួយមិនបានបញ្ចូលតំលៃតបម្រូវអោយ ចេញ message ថា "Please complete all information to log in"។
- បើបំពេញគ្រប់ ហើយតែ username ខុសពី "administrator" និង password ខុសពី "admin123" ត្រូវបង្ហាញ messageថា "username and password not correct, Please try again!"។
- បើត្រឹមត្រូវទាំងពីរខាងលើ ត្រូវ បង្ហាញ message ថា "Welcome to Administrator"
- ពេល user ចុច button clear ត្រូវលុបរាល់អក្សរក្នុង textbox ចោល។



Exercise 10

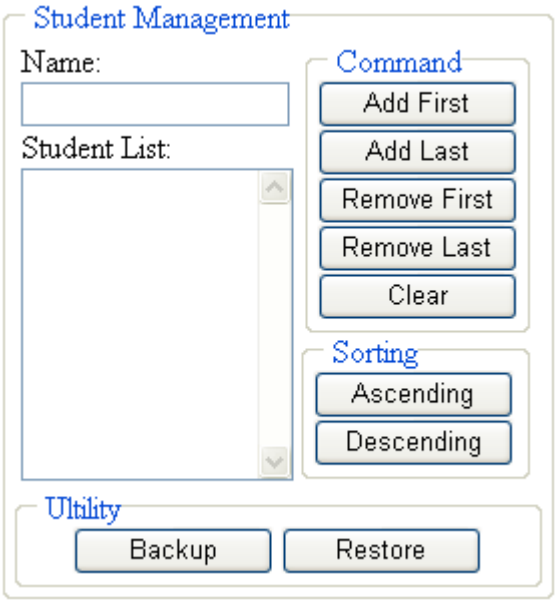
- ចូរអ្នកបង្កើត function ចំនួន ៥ សំរាប់ប្រមាណវិធី +,-,*,/,=
- បង្កើត textbox ចំនួន ៣ សំរាប់ដាក់ operand1, operand2, និង result
- ត្រូវប្រើ function តែពីរបុណ្ណោះ
- function ទី១ សំរាប់ប្រមាណវិធី +,-,*,/ ដែលមាន argument មួយចាំទទួល operator
- function ទី២ សំរាប់ប្រមាណវិធី = ដែតម្រូវអោយលទ្ធផលលើ Result



Exercise 11

- ចូរអ្នកបង្កើតនូវ Interface ដូចខាងក្រោមដោយប្រើ fieldset, legend, table, textarea, input button, input text និង CSS។
- រាល់ការងារទាំងអស់ត្រូវប្រើ array ជាដាច់ខាត ហើត្រូវបង្ហាញនៅលើ Student List (text area witdth=130px height=150px)។
- Add First បន្ថែមទៅផ្នែកតំបូងនៃ List
- Add Last បន្ថែមទៅផ្នែកខាងចុងនៃ List
- Remove First លុបផ្នែកតំបូងនៃ List
- Remove Last លុបផ្នែកខាងចុងនៃ List
- Clear លុបទិន្នន័យទាំងអស់
- Asscending តំរៀបទិន្នន័យតាមអក្ខរក្រមអង់គ្លេស
- Descending តំរៀបទិន្នន័យបញ្ច្រាស់ពីអក្ខរក្រមអង់គ្លេស
- Backup ត្រូវប្រើ array មួយទៀតធ្វើយ៉ាងណាអោយ array ទាំងពីរមានតំលៃដូចគ្នា។

- Restore យកតំលៃពី array ដែលបាន Backup ពីមុនមកដាក់ចូល arrayដើមវិញ



Appendix A: Key Words and Reserved Word

Keywords

break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	
delete	in	try	
do	instanceof	typeof	

Reserved Words

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

Appendix B: ASCII Code

ASCII CODE

0		30	-	60	<	90	Z	120	x	150	–	180	´	210	Ò	240	ð
1		31		61	=	91	[	121	y	151	—	181	µ	211	Ó	241	ñ
2		32		62	>	92	\	122	z	152	˜	182	¶	212	Ô	242	ò
3		33	!	63	?	93	]	123	{	153	™	183	·	213	Õ	243	ó
4		34	"	64	@	94	^	124	|	154	š	184	¸	214	Ö	244	ô
5		35	#	65	A	95	_	125	}	155	›	185	¹	215	×	245	õ
6		36	$	66	B	96	`	126	~	156	œ	186	º	216	Ø	246	ö
7		37	%	67	C	97	a	127	□	157	•	187	»	217	Ù	247	÷
8		38	&	68	D	98	b	128	€	158	ž	188	¼	218	Ú	248	ø
9		39	'	69	E	99	c	129	•	159	Ÿ	189	½	219	Û	249	ù
10		40	(	70	F	100	d	130	‚	160		190	¾	220	Ü	250	ú
11		41	)	71	G	101	e	131	*ƒ*	161	¡	191	¿	221	Ý	251	û
12		42	*	72	H	102	f	132	„	162	¢	192	À	222	Þ	252	ü
13		43	+	73	I	103	g	133	…	163	£	193	Á	223	ß	253	ý
14		44	,	74	J	104	h	134	†	164	¤	194	Â	224	à	254	þ
15		45	-	75	K	105	i	135	‡	165	¥	195	Ã	225	á	255	ÿ
16		46	.	76	L	106	j	136	ˆ	166	¦	196	Ä	226	â		
17		47	/	77	M	107	k	137	‰	167	§	197	Å	227	ã		
18		48	0	78	N	108	l	138	Š	168	¨	198	Æ	228	ä		
19		49	1	79	O	109	m	139	‹	169	©	199	Ç	229	å		
20		50	2	80	P	110	n	140	Œ	170	ª	200	È	230	æ		
21		51	3	81	Q	111	o	141	•	171	«	201	É	231	ç		
22		52	4	82	R	112	p	142	Ž	172	¬	202	Ê	232	è		
23		53	5	83	S	113	q	143	•	173		203	Ë	233	é		
24		54	6	84	T	114	r	144	•	174	®	204	Ì	234	ê		
25		55	7	85	U	115	s	145	‘	175	¯	205	Í	235	ë		
26		56	8	86	V	116	t	146	’	176	°	206	Î	236	ì		
27		57	9	87	W	117	u	147	“	177	±	207	Ï	237	í		
28		58	:	88	X	118	v	148	”	178	²	208	Ð	238	î		
29		59	;	89	Y	119	w	149	•	179	³	209	Ñ	239	ï		
19		49	1	79	O	109	m	139	‹	169	©	199	Ç	229	å		
20		50	2	80	P	110	n	140	Œ	170	ª	200	È	230	æ		
21		51	3	81	Q	111	o	141	•	171	«	201	É	231	ç		
22		52	4	82	R	112	p	142	Ž	172	¬	202	Ê	232	è		
23		53	5	83	S	113	q	143	•	173		203	Ë	233	é		
24		54	6	84	T	114	r	144	•	174	®	204	Ì	234	ê		
25		55	7	85	U	115	s	145	‘	175	¯	205	Í	235	ë		
26		56	8	86	V	116	t	146	’	176	°	206	Î	236	ì		
27		57	9	87	W	117	u	147	“	177	±	207	Ï	237	í		
28		58	:	88	X	118	v	148	”	178	²	208	Ð	238	î		
29		59	;	89	Y	119	w	149	•	179	³	209	Ñ	239	ï		

Special ASCII Code

8 : Backspace	9 : Tab
13 : Enter	16 : shift
17 : ctrl	18 : Alt
19 : Pause Break	20 : Caps Lock
32 : Space	33 : PageUp
34 : PageDown	35 : End Key
36 : Home Key	37 : Left Arrow Key
38 : Up Arrow Key	39 : Right Arrow Key
40 : Down Arrow Key	45 : Insert Key
46 : Delete Key	144 : Num Lock

Appendix C: JavaScript Version

JavaScript Versions

Browser Version	JavaScript Version
Netscape 2.x	1.0
Netscape 3.x	1.1
Netscape 4.0 – 4.0.5	1.2
Netscape 4.0.6 – 4.7x	1.3
Netscape 6.x, Mozilla 0.9	1.5
Firefox 1.5	1.6
Firefox 2.0	1.7
Firefox 3.0	1.8
Internet Explorer 3.x	JScript 1.0
Internet Explorer 4.x	JScript 3.0
Internet Explorer 5.x	JScript 5.0
Internet Explorer 5.5	JScript 5.5
Internet Explorer 6.x	JScript 5.6
Internet Explorer 7.x	JScript 5.6 (5.7 under Vista)

Reference

1. Professional JavaScript for Web Developers (Wrox) By Nicholas C. Zakas
2. JavaScript Pocket Reference 2nd Edition (O'Reilly) By David Flanagan
3. JavaScript 2.0: The Complete Reference, 2nd Edition(McGraw-Hill/Osborne) By Thomas Powell and Schneider