الفصل الرابع:

أساسيات لغة JavaScript

الكلمات المفتاحية

المتغيرات، أنماط البيانات، التعابير والعمليات، الإدخال والإخراج، الشرط، التكرار، الوظائف، المصفوفات، مطابقة النماذج.

الملخص

نستعرض في هذا الفصل أساسيات لغة البرمجة JavaScript.

الأهداف التعليمية

يتعرف الطالب في هذا الفصل على

- المتغيرات.
- الأنماط.
- العمليات.
- التعابير.
- الإدخال والإظهار.
- تعليمات التحكم.
- المصفوفات.
- الوظائف.
- مطابقة النماذج.

المخطط:

يضم فصل أساسيات لغة JavaScript 5 وحدات (Learning Objects) هي:

- أساسيات JavaScript
- تعليمات التحكم
- المصفوفات
- الوظائف Functions
- مطابقة النماذج

أساسيات JavaScript

الأهداف التعليمية:

- أساسيات البرمجة غرضية التوجه و JavaScript
- الخصائص العامة للشكل
- التعليمات
- الأنماط والعمليات والتعابير
- التصريح عن المتغيرات
- العمليات الرقمية
- تحويل الأنماط الضمني
- تحويل الأنماط الصريح
- الغرض Date
- الإدخال والإظهار
- الغرضwindow

مقدمة

ظهرت اللغة الخطاطية JavaScript عام 1996 نتيجة تعاون مثمر بين الشركتين Netscape و Sun.
وأصبحت اليوم معيار عالمي معتمد ISO-16262.
تتألف لغة JavaScript من ثلاثة أجزاء رئيسية:

- **قلب اللغة Core**

تحوي التعليمات الأساسية للغة

- **جهة الزبون Client Side**

تحوي مجموعة الأغراض التي تدعم التحكم بالمستعرض والتفاعل مع المستخدم (وهو الجزء الأكثر استخداماً من قبل مطوري الويب)

- **جهة المخدم Server Side**

تحوي مجموعة الأغراض التي يُمكن أن تتعامل مع مخدم الويب، كعمليات الوصول إلى قواعد البيانات مثلاً.

البرمجة غرضية التوجه و JavaScript

لا تدعم JavaScript البرمجة غرضية التوجه. أي أنها لاتحوي صفوف classes. بل تعمل أغراضها objects كأغراض وكنموذج للأغراض في آن واحد models of objects. كما لا تدعم JavaScript الوراثة Inheritanceوتعددية الأشكال Polymorphism.

يكون للأغراض في JavaScript مجموعة من الخصائص التي يُمكن أن تكون إما خاصية معطيات property أو طريقة method.

الخصائص العامة للشكل

- يُمكن تضمين الخططات مباشرة في ملف XHTML:

```
<script type="text/javaScript">
        -- JavaScript script -
    </script>
```

- أو (وهو الأفضل) وضع الخططات في ملف نصي مستقل وتضمينها باستخدام:

```
<script type="text/javaScript"
            src="myScript.js">
    </script>
```

- يُمكن وضع تعليمات اللغة ضمن تعليقات خاصة من الشكل التالي وذلك بهدف إخفاء التعليمات عن المتصفحات التي لاتدعم JavaScript:

```
<!--
-- JavaScript script -
//-->
```

المُعرّفات

- يجب أن تبدأ المعرفات Identifiers في JavaScript بحرف أو تحت السطر (_) أو إشارة الدولار $. بعدها يُمكن أن يحوي المعرف على حرف أو رقم أو تحت السطر أو دولار.
- لايوجد حد لطول المعرف.
- تكون javaScript حساسة لحالة الأحرف case sensitive.

الكلمات المفتاحية

تحوي اللغة على مجموعة من الكلمات المفتاحية:

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

التعليقات

يُمكن وضع التعليقات على سطر باستخدام // أو على أكثر من سطر باستخدام /* . . . . */.

التعليمات

يُستحسن وضع كل تعليمة على سطر وإنهائها بوضع فاصلة منقوطة (;). إذ أن مفسر اللغة interpreter يضع فاصلة منقوطة عند كل نهاية سطر لما يعتبره تعليمة. مما قد يقود لخطأ كما يُبين المثال التالي:

return x;

(حيث سيقوم المفسر بوضع فاصلة منقوطة بعد return، مما سيجعل وضع x غير قانوني).

الأنماط والعمليات والتعابير

الأنماط الأساسية

يكون للقيم أحد الأنماط الأساسية التالية:

Number, String, Boolean, Undefined, Null.

الأرقام والسلاسل النصية

تُخزن الأرقام باستخدام الفاصلة العائمة مع دقة مضاعفة. تُبين الأمثلة التالية أشكال صحيحة للأرقام:

72, 7.2, .72, 72., 7E2, 7e2, .7e2, 7.e2, 7.2E-2

تُحاط السلاسل النصية إما بإشارة تنصيص واحدة (') أو بإشارتي تنصيص("). يُمكن أن تحوي السلاسل النصية على محارف خاصة مثل \n و \t.

يُمكن استخدام المحرف (\) لإلغاء: (')

'You\'re the most freckly person I\'ve ever met'

كما يجب وضع (\) قبل كل (\) إذا كان من محارف السلسلة: "d:\\bookfiles"

لايوجد فرق بين السلسلة المحاطة بـ(') والمحاطة بـ ("). كما يُمكن التعبير عن السلسلة الفارغة بـ " أو "".

الأنماط الأساسية الأخرى

- يكون في النمط Boolean القيمتين true و false فقط. تنتج هذه القيم عادةً عن حساب تعبير منطقي.
- يكون في النمط Null قيمة وحيدة هي الكلمة المفتاحية null. وتُعامل كـ 0 عند معاملتها كرقم، وكـ false عند معاملتها كمتغير منطقي.
- يكون في النمط Undefined قيمة وحيدة هي undefined. وتُعامل كـ NaN عند معاملتها كرقم، وكـ false عند معاملتها كمتحول منطقي.
- يكون متغير undefined عندما يكون معرفاً ولم تُسند له قيمة.

فمثلاً إذا كتبنا التعليمات التالية:

```
<script type="text/javaScript">
   var a;
   var b = 10;
   b = b + a;
   document.write("a: ", a, "<br />");
   document.write("b: ", b, "<br />");
</script>
```

تكون النتيجة:

a: undefined
b: NaN

التصريح عن المتغيرات

تتميز JavaScript بأنها تقوم بتحديد نمط المتغير بشكل ديناميكي حسب القيمة المسنده له. كما يُمكن إسناد قيمة من أي نمط لنفس المتغير.

يُمكن التصريح عن متغير بشكل ضمني وذلك بإسناد قيمة له: a= 10;

أو بشكل صريح باستخدام الكلمة المفتاحية var:

```
var a,
sum = 0,
today = "Monday,"
flag = false;
```

العمليات الرقمية

توفر JavaScript العمليات الرقمية:

++, --, +, -, *, /, %

تراعي JavaScript أفضلية العمليات حيث تكون أفضلية *,/,% أعلى من +,-. وفي حال تساوي الأفضليات في تعبير يتمّ تطبيق العمليات من اليسار لليمين.

مثال:

يُبين المثال التالي كيفية تطبيق الأفضليات والتجميع:

```
<script type="text/javaScript">
       var a = 2,
           b = 4,
           c,
           d;
       c = 3 + a * b;
       // * is first, so c is now 11 (not 24)
       d = b / a / 2;
       // / association left, so d is now 1 (not 4)
       document.write("c: ", c, "<br />");
       document.write("d: ", d, "<br />");
</script>
```

يُمكن استخدام الأقواس لتحديد الأفضلية المطلوبة.

الغرض Math

يوفر الغرض Math مجموعة من الطرق على الأرقام مثل:

floor, round, max, min, cos, sin. . . ,

مثلاً:

Math.sin(x)

الغرض Number

- يوفر الغرض Number مجموعة من الخصائص ذات القيم الثابتة الرقمية:

MAX_VALUE, MIN_VALUE, NaN, POSITIVE_INFINITY, NEGATIVE_INFINITY, PI.

(مثلاً، تُعطي Number.Min_VALUE أصغر قيمة ممكنة).

- تُعيد عملية جبرية مع فيضان overflow القيمة NaN.
- تُستخدم الدالة isNaN() لاختبار أن متغير له القيمة NaN.
- للغرض Number الطريقة toString() لإرجاع الرقم كسلسلة نصية:

```
Var     price= 477,
Str_price;
. . .
Str_price = price.toString();
```

جمع السلاسل النصية

تُستخدم إشارة الجمع + لجمع السلاسل النصية:

```
<script type="text/javaScript">
var x = "Hello";
 x = x + " World";
 // x now is Hello World
document.write( x, "<br />");
</script>
```

تحويل الأنماط الضمني

تقوم JavaScript بمجموعة من التحويلات الضمنية بين الأنماط وفق ما يلي:

- إذا كانت العملية عملية جمع بين رقم وسلسلة نصية يتمّ تحويل الرقم إلى سلسلة نصية.
- إذا كانت العملية عملية حسابية (غير الجمع) يتمّ تحويل السلسلة النصية إلى رقم.
- إذا فشلت عملية تحويل السلسلة النصية إلى رقم تُعاد القيمة NaN.

مثال:

يُبين المثال التالي مختلف حالات التحويل الضمني:

```
<script type="text/javaScript">
var x, y, z, t;
 x = "August " + 2007;
 // x now is August 2007
document.write("x now is ", x, "<br />");
 y = 2007 + " August";
 // y now is 2007 August
document.write("y now is ", y, "<br />");
 z = 7 * "3";
 // z now is 21
document.write("z now is ", z, "<br />");
 t = "lo" * 3;
 // t now is NaN
document.write("t now is ", t, "<br />");
</script>
```

تحويل الأنماط الصريح

يُمكن طلب التحويل بين الأنماط بشكل صريح كما يلي:

- يُستخدم الباني String للحصول على سلسلة نصية.
- يُستخدم الباني Number للحصول على رقم.
- تُستخدم الطريقة toString() على رقم لتحويله إلى سلسلة نصية.
- يُمكن استخدام الدالة parseInt لتحويل سلسلة نصية إلى رقم صحيح.
- يُمكن استخدام الدالة parseFloat لتحويل سلسلة نصية إلى رقم عشري.

مثال:

يُبين المثال التالي مختلف حالات التحويل الصريح:

```
<script type="text/javaScript">
var str1 = String(33.33);
 // str1 now is "33.33"
document.write("str1 now is ", str1, "<br />");
 var num1 = 6.6;
 var str2 = num1.toString();
 //str2 now is "6.6"
document.write("str2 now is ", str2, "<br />");
 var num2 = Number(str1);
 // num2 now is 33.33
document.write("num2 now is ", num2, "<br />");
 var num3 = str1 - 0;
 // num3 now is 33.33
document.write("num3 now is ", num3, "<br />");
 var num4 = parseInt(str1);
document.write("num4 now is ", num4, "<br />");
 // num4 now is 33
 var num5 = parseFloat(str1);
 // num5 now is 33.33
document.write("num5 now is ", num5, "<br />");
</script>
```

خصائص وطرق السلاسل String

- للغرض String خاصية واحدة هي length وتعطي عدد الأحرف في سلسلة نصية.
- للغرض String مجموعة من الطرق أهمها:
  - charAt(number)
  - indexOf(One-character string)
  - substring(number1, number2)
  - toLowerCase()
  - toUpperCase()

كما تُبين الأمثلة:

```
var str="George";
str.length is 6
str.charAt(2) is 'o'
str.indexOf('r') is 3
str.substring(2, 4) is 'or'
str.toLowerCase() is 'george'
```

الطريقة typeof

تُعيد typeof نمط متغير كما تُبين الأمثلة التالية:

```
typeof("George") is string
typeof(33) is number
typeof(true) is Boolean
var a; typeof(a) is undefined
typeof(b) is undefined (b is a not defined var)
```

الإسناد

تُستخدم إشارة المساواة للإسناد:

```
a = a + 7;
a += 7;
```

الغرض Date

يُمكن التصريح عن متغير تاريخ يأخذ التاريخ والوقت الحالي:

var today = new Date();

يوفر الغرض Date مجموعة من الطرق:

toLocaleString, getDate, getMonth, getDay, getFullYear, getTime, getHours, getMinutes, getSeconds, getMilliseonds.

الإدخال والإظهار

يقوم مفسر JavaScript بتفيذ التعليمات حيثما يجدها ضمن المستند XHTML. وبالتالي فإن شاشة الإظهار الطبيعية لـ JavaScript هي نفسها شاشة إظهار مخرجات XHTML.

تُستخدم الطريقة write للغرض document بشكل أساسي لخلق خطاطة خرج. كما يُبين المثال التالي:

```
var a = 25;
document.write("The result is : <b> ", a, "</b> <br />");
```

حيث يكون الخرج:

The result is : **25**

لاحظ أن معامل الطريقة write يُمكن أن يحوي أي مؤثر XHTML.

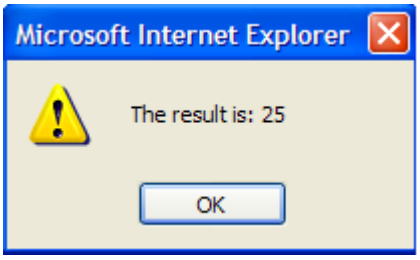
الغرض window

يوفر الغرض window ثلاثة طرق لإنشاء صناديق حوار بهدف التفاعل مع المستخدم.

- **الطريقة alert**

تقوم الطريقة alert بفتح نافذة حوارية وإظهار محتوى معاملها فيها.

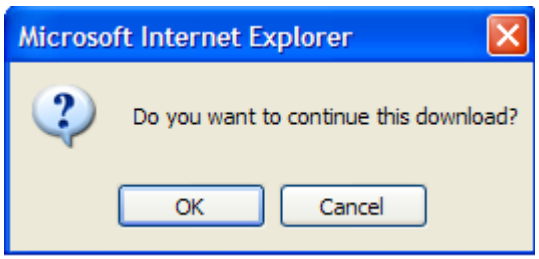
```
var a = 25;
alert("The result is: " + a + "\n");
```



- **الطريقة confirm**

تقوم الطريقة confirm بفتح نافذة حوارية مع زري OK و Cancel. تكون القيمة المرجعة true في حال نقر المستخدم على الزر OK، و false في حال نقره على الزر Cancel.

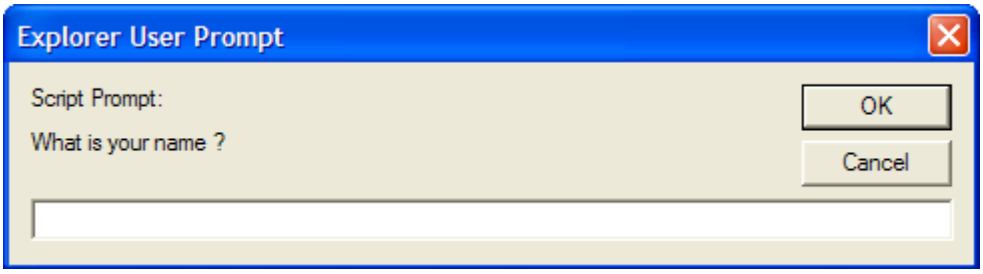
```
var question =
confirm("Do you want to continue this download?");
```



- **الطريقة prompt**

تقوم الطريقة prompt بإظهار نافذة حوارية تحوي صندوق نص للكتابة فيه. وتكون القيمة المرجعة هي محتوى هذا النص إذا نقر المستخدم على الزر OK، و null إذا نقر المستخدم على الزر Cancel.

```
a = prompt("What is your name ?", "");
```



مثال:

يقوم المثال التالي بحل معادلة من الدرجة الثانية بعد طلب حدودها من المستخدم:

```
<!DOCTYPE html >

<!-- roots.html
     Compute the real roots of a given quadratic
     equation. If the roots are imaginary, this script
     displays NaN, because that is what results from
     taking the square root of a negative number
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> Real roots of a quadratic equation </title>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
      var question=true;
      var a, b, c, root_part, denom, root1, root2;
      while (question)
      {
     // Get the coefficients of the equation from the user
      a = prompt("What is the value of 'a'? \n", "1");
      b = prompt("What is the value of 'b'? \n", "");
      c = prompt("What is the value of 'c'? \n", "");
      // Compute the square root and denominator of the result
      root_part = Math.sqrt(b * b - 4.0 * a * c);
      denom = 2.0 * a;
      // Compute and display the two roots
      root1 = (-b + root_part) / denom;
      root2 = (-b - root_part) / denom;
      if (isNaN(root1))
      {
       alert("No real roots !");
       question=confirm("Try another equation?");
      }
      else
      {
        document.write("The first root is: ", root1, "<br />");
        document.write("The second root is: ", root2, "<br />");
        question=false;
      }
      }
      // -->
    </script>
  </body>
</html>
```

تعليمات التحكم

الأهداف التعليمية

- التعرف على تعليمات التحكم

تعليمات التحكم

- تُشابه تعليمات JavaScript تعليمات لغة C++ و Java.
- تكون التعليمات المركبة تسلسل من التعليمات المحاطة بـ { }.

التعابير المنطقية

تكون نتيجة تقويم تعبير منطقي القيمة true أو القيمة false. يوجد ثلاثة أنواع من التعابير المنطقية:

- القيم الأساسية Primitive values
  - إذا كانت القيمة رقمية فهي تُعتبر true مالم تكن مساوية للصفر.
  - إذا كانت القيمة نصية تُعتبر true مالم تكن فارغة "" أو "0".
- التعابير العلائقية Relational Expressions
  - تستخدم العلاقات المعروفة ==, !=, <, >, =>, <=
  - في حال كون المعاملات من أنماط مختلفة يتمّ إجراء التحويل الضمني.
- التعابير المركبة Compound Expressions
  - يُمكن إنشاء تعابير مركبة من التعابير السابقة باستخدام العمليات المنطقية: &&(And), ||(Or), !(Not).

تعليمات الاختيار Selection Statements

التعليمة الشرطية if

يكون للتعليمة الشرطية if في JavaScript نفس الشكل في C. كما يُبين المثال التالي:

```
       if (a > b)
            document.write("a is greater than b <br />");
        else {
            a = b;
            document.write(" a was not greater than b <br />",
                           "Now they are equal <br /> ");}
```

التعليمة switch

تأخذ التعليمة switch الشكل:

```
switch (expression) {
            case value_1:
                // value_1 statements
            case value_2:
                // value_2 statements
            …
                [default:
                // default statements]
}
```

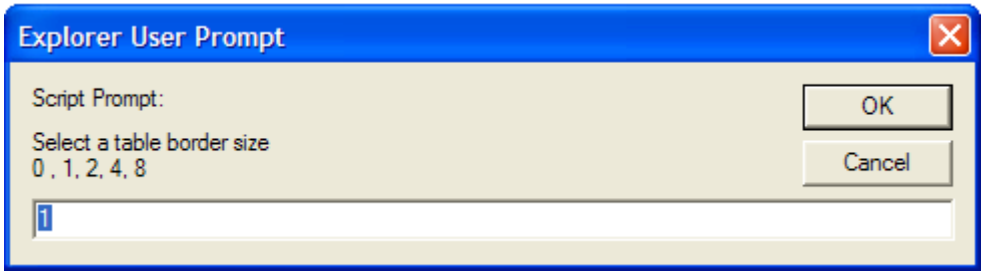
مثال:

يُبين المثال التالي استخدام switch حيث يتمّ الطلب من المستخدم إدخال عرض الإطار المطلوب لجدول:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> A switch statement </title>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
      var bordersize;
      bordersize = prompt("Select a table border size \n" +
                          "0 , 1, 2, 4, 8 \n","1" );
      switch (bordersize) {
        case "0": document.write("<table>");
                  break;
        case "1": document.write("<table border = '1'>");
                  break;
        case "4": document.write("<table border = '4'>");
                  break;
        case "8": document.write("<table border = '8'>");
                  break;
        default:  document.write("Error - invalid choice: ",
                                 bordersize, "<br />");
      }

      document.write("<caption> 2003 NFL Divisional",
                     " Winners </caption>");
      document.write("<tr>",
                     "<th />",
                     "<th> American Conference </th>",
                     "<th> National Conference </th>",
                     "</tr>",
                     "<tr>",
                     "<th> East </th>",
                     "<td> New England Patriots </td>",
                     "<td> Philadelphia Eagles </td>",
                     "</tr>",
                     "<tr>",
                     "<th> North </th>",
                     "<td> Baltimore Ravens </td>",
                     "<td> Green Bay Packers </td>",
                     "</tr>",
                     "<tr>",
                     "<th> West </th>",
                     "<td> Kansas City Chiefs </td>",
                     "<td> St. Louis Rams </td>",
                     "</tr>",
                     "<tr>",
                     "<th> South </th>",
                     "<td> Indianapolis Colts </td>",
                     "<td> Carolina Panthers </td>",
                     "</tr>",
                     "</table>");
      // -->
    </script>
  </body>
</html>
```

يبدأ التنفيذ بالطلب من المستخدم إدخال عرض الحدود المطلوب:



ومن ثم يتمّ إظهار الجدول بالعرض المحدد:

2003 NFL Divisional Winners

	American Conference	National Conference
East	New England Patriots	Philadelphia Eagles
North	Baltimore Ravens	Green Bay Packers
West	Kansas City Chiefs	St. Louis Rams
South	Indianapolis Colts	Carolina Panthers

تعليمات التكرار

توفر لغة JavaScript التعليمتين الأساسيتين للتكرار while و for. يكون للتعليمة while الشكل:

```
while ( control expression)
        {
            Statements
}
```

أو الشكل:

```
do
        {
            statements
        }
        while ( control expression)
```

يكون للتعليمة for الشكل:

```
for (initial expression; control expression; increment expression)
        {
           statements
      }
```

المصفوفات Arrays

الأهداف التعليمية

- المصفوفات

المصفوفات Arrays

يتمّ تعريف المصفوفات في JavaScript إما باستخدام التعليمة new أو بإسناد عناصر المصفوفة مباشرة.
كما تُبين الأمثلة التالية:

```
var myList1 = new Array(24);
var myList2 = ["bread", 99, true];
```

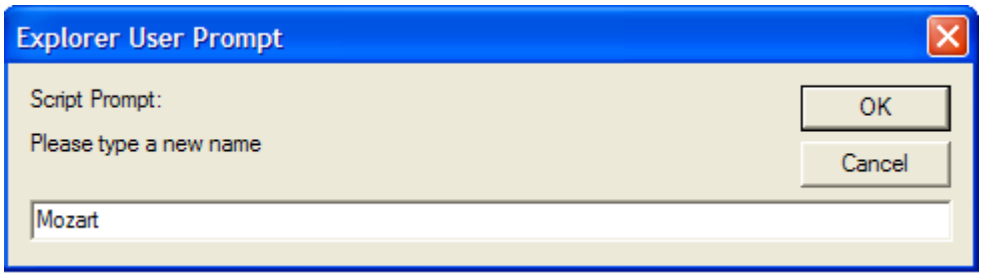
- يكون طول المصفوفة length هو فهرس آخر عنصر فيها زائد 1.
- يُمكن تغيير طول المصفوفة بإسناد قيمة إلى الخاصية length: myList.length = 150;

مثال:

يوضح المثال التالي استخدام المصفوفات. نصرح في البداية عن مصفوفة تحوي بعض الأسماء مرتبة أبجدياً.
نطلب من المستخدم إدخال اسم جديد ونقوم بإدراجه في المصفوفة وبحيث نحافظ على الترتيب الأبجدي.

```
<!DOCTYPE html >
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Name list </title>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
// The original list of names
var name_list= new Array("Al", "Betty", "Kasper",
                                  "Michael", "Roberto", "Zimbo");
var new_name, index, last;
// Loop to get a new name and insert it
      while (new_name = prompt("Please type a new name", "")) {
      // Loop to find the place for the new name
        last = name_list.length - 1;
           while (last >= 0 && name_list[last] > new_name) {
                  name_list[last + 1] = name_list[last];
                  last--;
          }
     // Insert the new name into its spot in the array
         name_list[last + 1] = new_name;
    // Display the new array
        document.write("<p><b>The new name list is:</b> ",
                       "<br />");
        for (index = 0; index < name_list.length; index++)
          document.write(name_list[index], "<br />");
        document.write("</p>");
      } //** end of the outer while loop
      // -->
    </script>
  </body>
</html>
```

يبدأ البرنامج بطلب اسم جديد من المستخدم:



ثم يظهر المصفوفة الجديدة بعد إضافة الاسم الجديد في مكانه الصحيح:

The new name list is:
Al
Betty
Kasper
Michael
Mozart
Roberto
Zimbo

يوجد مجموعة من الطرق للتعامل مع المصفوفات:

- concat لوصل مصفوفة مع أخرى.
- join لتشكيل سلسلة نصية من عناصر المصفوفة مع فصلها بفاصل محدد.
- reverse لعكس عناصر المصفوفة.
- slice للحصول على جزء من المصفوفة.

مثال:

يُبين المثال التالي بعض طرق التعامل مع المصفوفات:

```
<script type="text/javaScript">
  a = new Array("a", "b", "c", "d");
  n = new Array(1, 2, 3);
  an = a.concat(n);
  document.write("a concat n= ", an, "<br />");
  document.write("a.join(',') = ", a.join(","), "<br />");
  document.write("a.slice(1,3) =", a.slice(1, 3), "<br />");
  document.write("a.reverse() =", a.reverse(), "<br />");
<script>;
```

حيث يكون الخرج:

```
a concat n= a,b,c,d,1,2,3
a.join(',') = a,b,c,d
a.slice(1,3) =b,c
a.reverse() =d,c,b,a
```

Functions الوظائف

الأهداف التعليمية

- الوظائف

الوظائف Functions

- يُشبه شكل الوظائف في JavaScript شكل الوظائف في لغة C.
- يتمّ التصريح عن الوظائف ضمن المؤثر <head>.
- لايتمّ التحقق عند استدعاء الوظيفة من نمط المعاملات الممرة ولا من عددها. حيث يتم تجاهل المعاملات الزائدة. أما المعاملات الناقصة فتُعتبر undefined.
- يتمّ إرسال المعاملات عبر مصفوفة arguments يُمكن الحصول على طولها من الخاصية length.

مثال:

يُبين المثال التالي استخدام المصفوفة arguments:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Parameters </title>
    <script type = "text/javascript">
          <!--
   // Function params
   // Parameters: two named parameters and one unnamed
   // parameter, all numbers
   // Returns: nothing
   function params(a, b) {
           document.write("Function params was passed ",
               arguments.length, " parameter(s) <br />");
           document.write("Parameter values are: <br />");
     for (var arg = 0; arg < arguments.length; arg++)
             document.write(arguments[arg], "<br />");
    document.write("<br />");
           }
         // -->
       </script>
     </head>
     <body>
   <script type = "text/javascript">
         params("Mozart");
         params("Mozart", "Beethoven");
         params("Mozart", "Beethoven", "Tchaikowsky");
   </script>
     </body>
   </html>
```

حيث يكون الخرج:

```
Function params was passed 1 parameter(s)
Parameter values are:
Mozart

Function params was passed 2 parameter(s)
Parameter values are:
Mozart
Beethoven

Function params was passed 3 parameter(s)
Parameter values are:
Mozart
Beethoven
Tchaikowsky
```

مطابقة النماذج

الأهداف التعليمية

- مطابقة النماذج

مطابقة النماذج Pattern Matching

توفر لغة JavaScript طرق قوية لمطابقة النماذج اعتماداً على التعابير المنتظمة Regular Expressions.

المحارف والمحارف المترفعة

يوجد نوعين من المحارف في النموذج:

- المحارف العادية وهي المحارف التي تتطابق مع نفسها.
- المحارف المترفعة وهي المحارف التي لها معنى خاص ولاتتطابق مع نفسها:

. ? + * $ ^ { } [ ] ( ) | \ (يُمكن جعل محرف مترفع يُعامل كمحرف عادي بسبقه بـ \ ).

تُعتبر الطريقة search على الغرض String من أبسط الطرق لمطابقة النماذج. يكون النموذج معامل الدخل لهذه الطريقة، وتُعيد موقع بداية النموذج في السلسلة في حال وجوده (تُفهرس الأحرف اعتباراً من الصفر)، أو – 1 في حال عدم وجوده. كما يُبين المثال التالي:

```
var str = "Rabbits are furry";
 var position = str.search(/bits/);
 if (position >= 0)
     document.write("'bits' appears in position",
                                        position, "  <br />");
 else
     document.write(" 'bits' does not appear in str <br />");
```

الجواب: 'bits' appears in position 3

المحارف والمحارف المترفعة

- يُطابق المحرف المترفع (.) أي محرف عدا السطر الجديد. فمثلاً، يُطابق النموذج /.snow/ كل من snowy, snowe, snowd. (لمطابقة نقطة في سلسلة نصية يجب سبق النقطة في النموذج بـ \، فمثلاً، يُطابق النموذج \3\.4\ السلسلة النصية 3.4 ولايطابق 374.
- عند وضع مجموعة من الأحرف ضمن [ ]. فهذا يعني أن المطابقة يجب أن تتم مع أحد هذه الحروف. فمثلاً يُطابق النموذج /n[oi]/ كل من on و in.
- يُمكن استخدام المحرف (–) لتعيين مجال من القيم. فمثلاً، /[a–h]/ تعني أي محرف بين a و h.
- يُمكن استخدام المحرف (^) لعكس المحارف المعينة. فمثلاً /[^abc]/ تعني أي محرف ماعدا الأحرف a, b, c.

المحارف والمحارف المترفعة

يوجد بعض الصفوف المعرفة مسبقاً لبعض النماذج الأكثر استخداماً:

Name	Equivalent Pattern	Matches
\d	[0-9]	a digit
\D	[^0-9]	not a digit
\w	[A-Za-z_0-9]	a word character
\W	[^A-Za-z_0-9]	not a word character
\s	[\r\t\n\f]	a whitespace character
\S	[^ \r\t\n\f]	not a whitespace character

فمثلاً:

- يُطابق النموذج /\d\.\d\d/ أي رقم تليه نقطة يليها رقمين.
- يُطابق النموذج /\D\d\D/ رقم واحد.

يُمكن في العديد من الحالات تحديد تكرار معين:

Quantifier	Meaning
{n}	exactly n repetitions
{m,}	at least m repetitions
{m, n}	at least m but not more than n repetitions

فمثلاً يُطابق النموذج /xy{4}z/ السلسلة xyyyyz.

- يُستخدم المحرف (*) لتحديد صفر أو أي عدد من التكرارات.
- يُستخدم المحرف (+) لتحديد تكرار واحد أو أكثر.
- يُستخدم المحرف (?) لتحديد صفر أو تكرار واحد.

فمثلاً، يُطابق النموذج /x?x*y+z/ سلسلة محرفية تبدأ بعدد من x (أو ولا x) يليها تكرار أو أكثر لـ y ، يليها z واحدة (أو ولا z).

يُطابق النموذج /\d+\.\d*/ رقم أو أكثر يليه نقطة يليها عدد من الأرقام (أو ولا رقم).

يُبين المثال التالي استخدام مطابقة النماذج لاختبار شكل سلسلة نصية:

```
<!DOCTYPE html >
<!-- forms_check.html
     A function tst_phone_num is defined and tested.
     This function checks the validity of phone
     number input from a form
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Phone number tester </title>
    <script type = "text/javascript">
      <!--
/* Function tst_phone_num
    Parameter: A string
    Result: Returns true if the parameter has the form of a legal
            seven-digit phone number (3 digits, a dash, 4 digits)
    */
function tst_phone_num(num) {
        var ok = num.search(/\d{3}-\d{4}/);
        if (ok == 0)
          return true;
        else
          return false;
      }  // end of function tst_phone_num
      // -->
    </script>
  </head>
<body>
    <script type = "text/javascript">
      <!--
 var tst = tst_phone_num("444-5432");
 if (tst)
   document.write("444-5432 is a legal phone number <br />");
 else
    document.write("Error in tst_phone_num <br />");
 tst = tst_phone_num("444-r432");
 if (tst)
    document.write("Error in tst_phone_num <br />");
 else
    document.write("444-r432 is not a legal phone number <br />");
 tst = tst_phone_num("44-1234");
 if (tst)
    document.write("Error in tst_phone_num <br />");
 else
    document.write("44-1234 is not a legal phone number <br /");
// -->
</script>
</body>
</html>
```

حيث يكون الخرج:

444-5432 is a legal phone number
444-r432 is not a legal phone number
44-1234 is not a legal phone number

تحديد الموقع

يُمكن استخدام المحرف (^) لتحديد أن موقع النموذج يجب أن يكون بداية السلسلة. أو المحرف ($) لتحديد أن النموذج يجب أن يكون نهاية السلسلة. فمثلاً، يُطابق النموذج /gold$/ السلسلة "I like gold" بينما لا يُطابق "golden". كذلك يُطابق النموذج /^pearl/ السلسلة "pearls are pretty" ولايُطابق "My pearls are pretty".

تعديل النماذج

يُمكن استخدام المحرف (i) لطلب تجاهل حالة الأحرف. فمثلاً، يُطابق النموذج /ok/i كل من OK, Ok, ok, oK.

الطريقة replace

يُمكن استخدام الطريقة replace لاستبدال سلسلة جزئية بأخرى. كما يُمكن استخدام المحرف (g) لطلب الاستبدال لكل ظهور للسلسلة الجزئية.

مثال:

```
var str = "Some rabbits are rabid";
str = str.replace(/rab/g, "tim");
document.write("str is ",str ,"<br />");
```

تصبح str مساوية إلى "Some timbits are timid".

الطريقة match

تُستخدم الطريقة match لإرجاع مصفوفة من السلاسل الجزئية المطابقة للنموذج.

مثال:

```
var str = "My 3 kings beat your 2 aces";
var matches = str.match(/[ab]/g);
document.write("matches is ", matches ,"<br />");
```

تصبح قيمة matches مساوية إلى b,a,a.

الطريقة split

تقوم الطريقة split بتجزئة السلسلة إلى سلاسل جزئية.

مثال:

```
var str = "red,green,blue";
var colors = str.split(",");
document.write("colors is ", colors,"<br />");
```

تصبح قيمة colors مساوية إلى [red, green , blue].