القسم الأول- التقنيات من جهة الزبون Client Side

الفصل الرابع–أساسيات لغة البرمجة جافاسكريبت

JavaScript

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 وتحوي التعليمات الأساسية للغة.
- جهة الزبون Client Side وتحوي مجموعة الأغراض التي تدعم التحكم بالمستعرض والتفاعل مع المستخدم (وهو الجزء الأكثر استخداماً من قبل مطوري الويب).
- جهة المخدم Server Side وتحوي مجموعة الأغراض التي يُمكن أن تتعامل مع مخدم الويب، كعمليات الوصول إلى قواعد البيانات مثلاً.

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:

await	**break**	**case**	**catch**	**class**
const	**continue**	**debugger**	**default**	**delete**
do	**else**	**enum**	**export**	**extends**
false	**finally**	**for**	**function**	**if**
implements	**import**	**in**	**instanceof**	**interface**
let	**new**	**null**	**package**	**private**
protected	**public**	**return**	**super**	**switch**
static	**this**	**throw**	**try**	**true**
typeof	**var**	**void**	**while**	**with**
yield				

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:

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

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;
```

- كما يمكن تعريف المتغيرات باستخدام let:

```
<script>
        let carName = "Volvo";
        let person  = "John Doe", carName = "Volvo", price = 200;
</script>
```

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.

يُبين المثال التالي كيفية تطبيق الأفضليات والتجميع:

```
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)
```

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:

```
var x = "Hello";
x = x + " World";
// x now is Hello World
```

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.

يُبين المثال التالي مختلف حالات التحويل الضمني:

```
var x, y, z, t;
x = "August " + 2007;
// x now is August 2007
y = 2007 + " August";
// y now is 2007 August
z = 7 * "3";
// z now is 21
t = "lo" * 3;
// t now is NaN
```

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.

يُبين المثال التالي مختلف حالات التحويل الصريح:

```
 var str1 = String(33.33);
 // str1 now is "33.33"
 var num1 = 6.6;
 var st22 = num1.toString();
 //str2 now is "6.6"
var num2 = Number(str1);
 // num2 now is 33.33
var num3 = str1 - 0;
```

```
 // num3 now is 33.33
var num4 = parseInt(str1);
// num4 now is 33
var num5 = parseFloat(str1);
// num5 now is 33.33
```

خصائص و طرق 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.

- تقوم الطريقة alert بفتح نافذة حوارية وإظهار محتوى معاملها فيها.

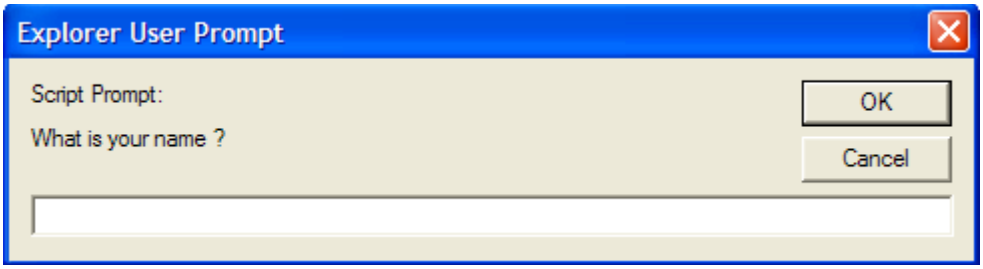
```
var a = 25;
alert("The result is: " + a + "\n");
```

- 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?");
```

- 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 ?", "");
```



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

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]
    }
```

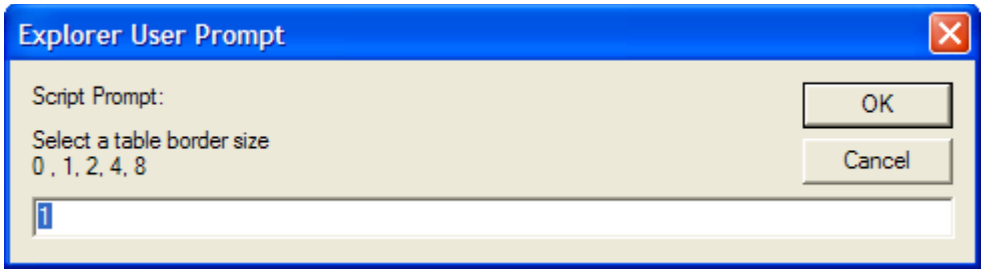
يُبين المثال التالي استخدام switch حيث يتمّ الطلب من المستخدم إدخال عرض الإطار المطلوب لجدول:

```
<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 "2": document.write("<table border = '2'>");
                  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

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:

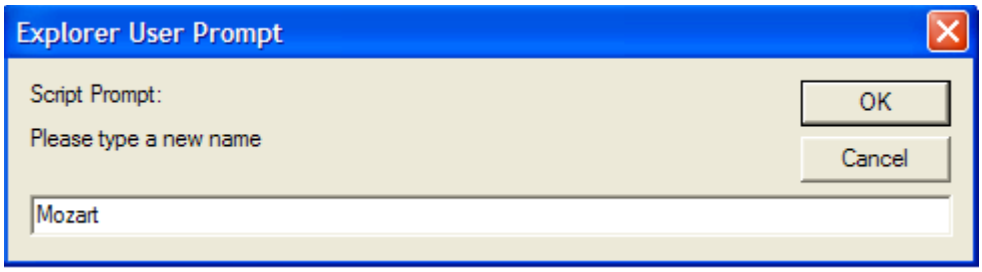
myList.length = 150;

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.

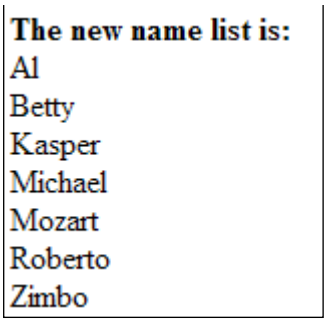
```
<!DOCTYPE html >
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Name list </title>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
// The original list of names
var name_list= new Array("Al", "Betty", "Kasper",
                                  "Michael", "Roberto", "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:



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.

يُبين المثال التالي بعض طرق التعامل مع المصفوفات:

```
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 />");
```

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
```

الوظائف Functions

- يُشبه شكل الوظائف في JavaScript شكل الوظائف في لغة C.
- لايتمّ التحقق عند استدعاء الوظيفة من نمط المعاملات الممررة 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. 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
```

البرمجة غرضية التوجه و JavaScript

تطورت جافاسكريبت وأصبحت لغة غرضية التوجه OOP منذ عام 2015، وتضمنت منذ ذلك الوقت مفاهيم التجريد Abstracting والوراثة Inheritance وتعددية الأشكال Polymorphism.

يبين المثال التالي تعريف الصف Employee مع باني Constructor ووظيفة Method. ومن ثم نقوم بإنشاء موظفين e1,e2 من هذا الصف ونستدعي الوظيفة detail.

ملاحظة: قد لا تدعم بعض المتصفحات البرمجة غرضية التوجه.

```
<!DOCTYPE html>
<html>
<body>

    <script>
        //Declaring class
        class Employee
          {
        //Initializing an object
            constructor(id,name)
            {
```

```
            this.id=id;
            this.name=name;
        }
    //Declaring method
        detail()
        {
            document.writeln(this.id+" "+this.name+"<br>")
        }
      }
    //passing object to a variable
    var e1=new Employee(101,"Martin Roy");
    var e2=new Employee(102,"Duke William");
    e1.detail(); //calling method
    e2.detail();
</script>

</body>
</html>
```

وتكون نتيجة التنفيذ:

101 Martin Roy
102 Duke William

كما يبين المثال التالي مفهوم الوراثة والتي هي مشابهة إلى لغات أخرى كالجافا والـ C#:

```
<!DOCTYPE html>
<html>
<body>

    <script>
        //Declaring class
        class Employee
          {
        //Initializing an object
            constructor(id,name)
            {
              this.id=id;
              this.name=name;
            }
        //Declaring method
            detail()
```

```
            {
                document.writeln(this.id+" "+this.name+"<br>")
            }
          }
        //passing object to a variable
        var e1=new Employee(101,"Martin Roy");
        var e2=new Employee(102,"Duke William");
        e1.detail(); //calling method
        e2.detail();

        // Creating a new class from the parent
        class info extends Employee {
            constructor(name, empid, salary) {
                // Chain constructor with super
                super(name, empid);

                // Add a new property
                this.salary = salary;
            }
        }

        var e3 = new info(103, "Salem Ahmad", 150000);
        e3.detail();
    </script>

</body>
</html>
```

حيث تكون نتيجة التنفيذ:

101 Martin Roy
102 Duke William
103 Salem Ahmad