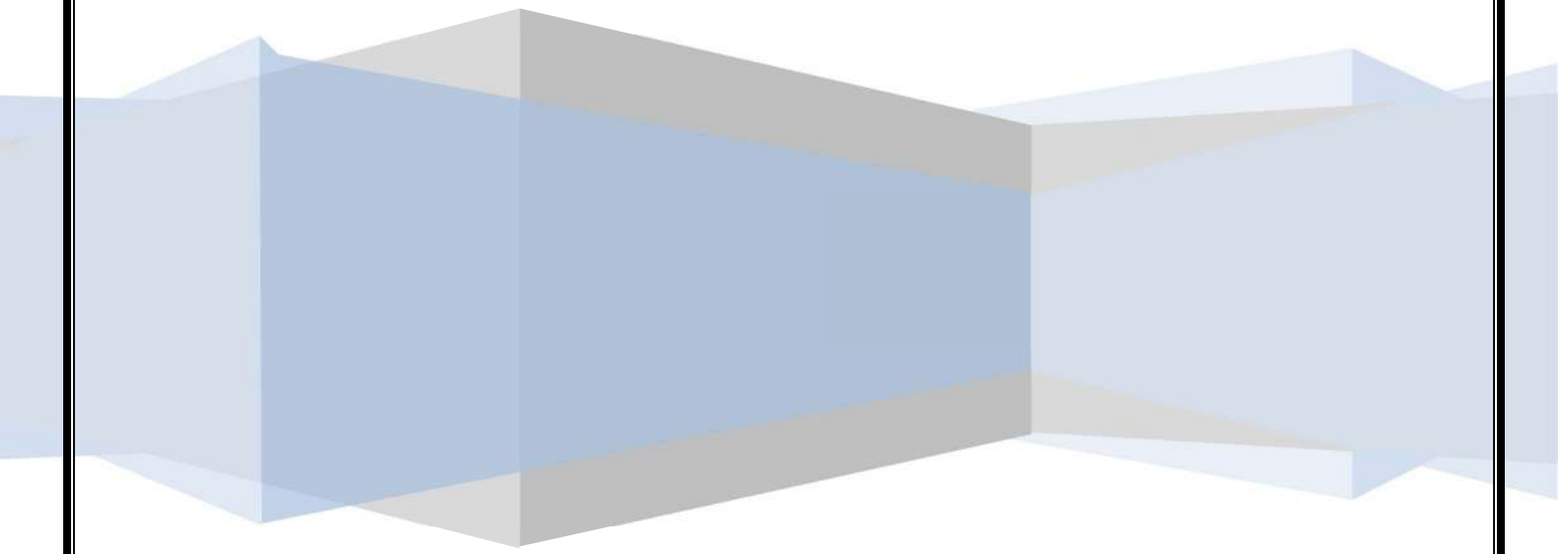
ملخص كورس JavaScript

للبشمهندس أسامة الزيرو

وترجمة من موقع w3schools



مقدمة:

لغة JavaScript من ابتكرها هو Brendan Each
وكان يعمل في شركة Netscape

وابتكر اللغة كإضافة للغة html وكانت تعمل في

متصفح Netscape Navigator
الإصدار الثاني

وكان الاسم الخاص بالجافا سكربت في البداية
هو اسم mocha ومع صدور الإصدارات الأولية للجافا السكربت

حصلت على اسم Live Script
ثم أخيرا حصلت على اسم JavaScript
وهناك فرق شاسع بين لغة Java ولغة JavaScript

لغة JavaScript من لغات interpreted Language
interpreted يعني لغة بتحتاج لمترجم علشان يترجم أكواد اللغه علشان تتعامل مع الكمبيوتر
والمترجم هو Browser أي المتصفح أي كان نوعه.

ولغة JavaScript تدعم البرمجة الكائنية وهي Object Oriented Programming
وهي OOP

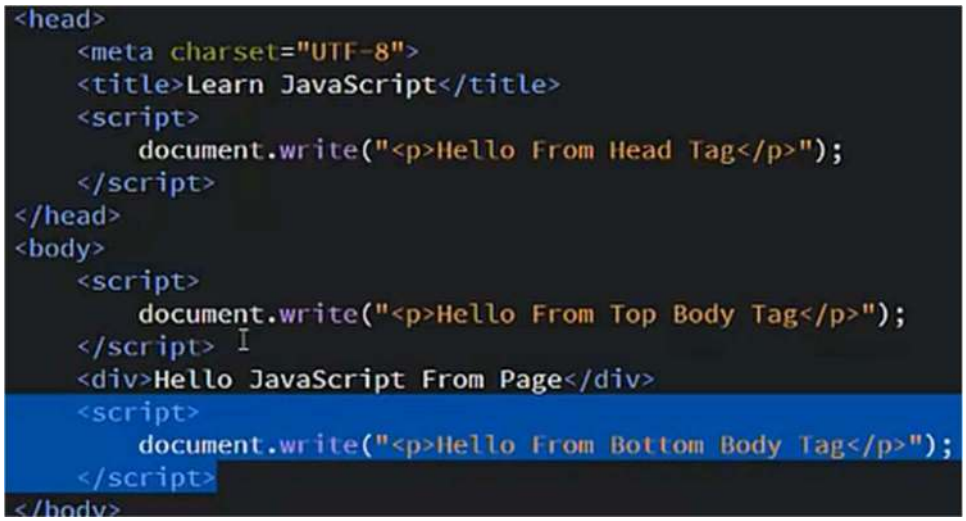
معظم استخدام لغه JavaScript بيكون مع الكلاين سايت مع المتصفح يعني بعيد عن السيرفر
يعني تقوم بتغيير محتوى المتصفح يتجعلك تتحكم في المتصفح
وحاليا الجافا تتعامل مع السيرفر سايت عن طريقة لغه النود

وتم دعم لغه الجافا سكريبت من قبل شركة مايكروسوفت في عام ٩٦ في متصفح انترنت
إكسبلورر الإصدار الثالثة

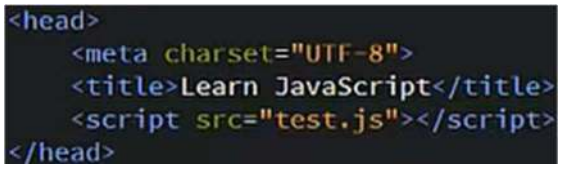
كود الجافا اسكربت

يتم وضع كود الجافا اسكربت <script> في أخر مكان في ال head

ويمكن وضع كود الجافا سكربت في أي مكان في ال body كذلك



والصحيح وكما يفضل المبرمجون هو وضع كود الجافا سكربت في ملف خارجي وليس داخل ملف html



ويضاف بهذه الطريقة وأمتداده js

وينصح كافة المبرمجين بوضع ملف الجافا اسكربت في أخر مكان في body

ملف الجافا سكربيت <script></script> يكتب في <head> بعد ال <style>
ويمكن كتابته في <body>

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Learn JavaScript</title>
        <script>
            document.write("<p>Hello From Head Tag</p>");
        </script>
    </head>
    <body>
        <script>
            document.write("<p>Hello From Head Tag</p>");
        </script>
        <div>Hello JavaScript From Page</div>
    </body>
</html>
```

والصحيح والأفضل هو وضع أكواد JavaScript في ملف منفصل خارجي ، كي تكون مفصوله عن أكواد html، وكذلك يكون سهل للقرأة وسهل في استخراج المشكلات والشئ الثالث موضوع الكاش يسرع من تصفح الموقع عندما يكون في ملف خارجي

كود الجافا السكربت الذي يكتب خارج صفحة html

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Learn JavaScript</title>
        <script src="test.js"></script>
    </head>
    <body>
        <div>Hello JavaScript From Page</div>
    </body>
</html>
```

ننشئ ملف جديد بإمتداد (js)
ونكتب في ال <head>

<script scr="test.js"></script>

وأفضل مكان لوضع ملف JavaScript والذي ينصح به المبرمجين هو قبل قفلة وسم body closetag body لل

مثال/

<script scr="test.js"></script>

JavaScript Syntax

بناء الجافا سكريبت

JavaScript Comments

لكتابة comment في الجافا اسكربت نضيف // double Slashes

```
// Simple Function

document.getElementById("test").innerHTML = "Hello JavaScript From Js File"; // Simple Function
```

```
// Single Line Comment

/*
    Multi Line Comment Line 1
    Multi Line Comment Line 2
*/

/*
*** Version 1.0
*** This Is Test Comment
*** This Is Line 2
*** This Is Last Line
*/

/*
```

لكتابة comment

يوجد **single line comment** وهذا في حالة كان الكومنت سطر واحد نكتب //

#####

####

ويوجد **Multi line comment**

وهذا في حالة أن comment أكثر من سطر فنكتب /* ### */

```
/*global console*/

// Some Vars For Bla Bla Bla

// Single Line COmment

/*
    Multi Line Comment Line 1
    Multi Line Comment Line 2
*/
```

JavaScript is Case Sensitive

جافا سكريبت حساس لحالة الأحرف

جميع معرفات جافا سكريبت حساسة لحالة الأحرف.

حروف ال JavaScript حساسة فالحروف الكابتل لا تساوي الحروف الصمول

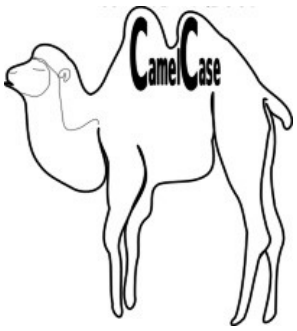
JavaScript and Camel Case

جافا سكريبت وحالة الجمل

Upper Camel Case (Pascal Case):حالة العلبة العلوية حالة باسكال

FirstName, LastName, MasterCard, InterCity.

وفيها يتم كتابة أول حرف من الكلمة الأولى وأول حرف من الكلمة الثانية وهكذا كابتل



Lower Camel Case : انخفاض حالة الجمل

وفيها يتم كتابة أو حرف من الكلمة كابتل كل كلمة يبدأ الحرف الأول منها بكابتل

firstName, lastName, masterCard, intercity

Variables : المتغيرات

يعرفها موقع w3 ب: هي حاويات لتخزين قيم البيانات

و يجب تحديد جميع متغيرات جافا سكريبت بأسماء فريدة.

هذه الأسماء الفريدة تسمى المعرفات

القواعد العامة لإنشاء أسماء للمتغيرات

١ : يمكن أن تحتوي الأسماء على أحرف ، وأرقام ، وشرطات سفلية ، وعلامات دولار$

٢ : يجب أن تبدأ الأسماء بحرف و يمكن أن تبدأ الأسماء أيضًا ب $

٣ : الأسماء حساسة لحالة الأحرف فـ حرف Y كابتل ليس هو حرف y صمول

٤ : الكلمات المحجوزة مثل (كلمات JavaScript الأساسية) لا يمكن استخدامها كأسماء

The Assignment Operator

مشغل الإحالة

في JavaScript ، تعد علامة المساواة (=) عامل تشغيل "التعيين" ، وليس عامل التشغيل "مساوٍ."

تتم كتابة عامل التشغيل "مساوٍ" مثل == في . JavaScript

Data Type : أنواع البيانات

أنواع بيانات الجافا سكربت

١ : يمكن لمتغيرات جافا سكريبت الاحتفاظ بأرقام مثل ١٠٠ والقيم النصية مثل "John Doe".

٢ : في البرمجة ، تُعرف القيم النصية بالسلاسل النصية

٣ : تتم كتابة السلاسل داخل علامات اقتباس مزدوجة " " أو مفردة ' ' . تتم كتابة الأرقام بدون علامات اقتباس.

٤ : إذا وضعت رقمًا بين علامتي اقتباس ، فسيتم التعامل معه كسلسلة نصية.

مثال :

Example

```
var pi = 3.14;
var person = "John Doe";
var answer = 'Yes I am!';
```

Declaring (Creating) JavaScript Variables

التصريح (انشاء) متغيرات جافا سكربت

يأخذ متغير جافا اسكربت هذه الكلمة المحجوزه له وهي Var

نكتب في JavaScript **var** كي نعرفه أننا سوف نكتب متغير

Value = undefined

```
var myPrice; // Undefined
```

المتغير المُعلَن بدون قيمة سيكون له قيمة غير محددة.

Undefined يعني مالهوش قيمة

لا يصح بدء المتغير برقم ولكن ممكن بدء المتغير ب
start with letters, underscore, $

```
/*
    Start With Letters, Underscore, $
*/
```

ومعناها

أننا نستطيع كتابة المتغيير يبدأ بحرف Letters

أو يبدأ ب Underscore وشرطات سفلية

أو بعلامة $ دولار سايز

خطأ : ولكن لا يصح بداية المتغير برقم فهذا خطأ

```
var 1osama
```

مثال

Var myprice = 100;

```
var myPrice = 100;
```

Var = JavaScript Variable Keyword : كلمة مفتاحية محجوزه للمتغير في الجافا سكربت

Mayprice = Variable Name : اسم المتغير

(=) = Assignment Operator لتعيين قيمة للمتغير

100 = Varible Value : قيمة المتغير

```
document.getElementById("price").innerHTML = myPrice;
```

ومعناها اطبع لي id ولابد من تواجد id في html فعلاً حتى يقوم بطباعته

```
<body>
 <div id="price"></div>
 <script src="testjava.js"></script>
</body>
```

مثال على المتغير واستخدام العمليات الحسابيه فيه

ونلاحظ:

يمكنك الإعلان عن العديد من المتغيرات في بيان واحد.

ويمكن للإعلان أن يمتد على عدة أسطر ولكن نفصل بين المتغيرات ب **comma**

```
var // My Product Prices

    myOldPrice = 2000,

    myNewPrice = 900,

    myDiscount = myOldPrice - myNewPrice + 500; // 2000 - 900 + 500 = 1600

document.getElementById("price").innerHTML = myDiscount;
```

مثال أخر

```
var mainPrice = 500,

    mySmallDiscount = 50,

    myMediumDiscount = 100,

    myBigDiscount = 250;

document.getElementById("price").innerHTML = mainPrice;

document.getElementById("product1").innerHTML = mainPrice - mySmallDiscount; // 500 - 50 = 450

document.getElementById("product2").innerHTML = mainPrice - myMediumDiscount; // 500 - 100 = 400

document.getElementById("product3").innerHTML = mainPrice - myBigDiscount; // 500 - 250 = 250
```

JavaScript Data Types

أنواع بيانات جافا اسكربت

يمكن لمتغيرات جافا سكريبت الاحتفاظ بالعديد من أنواع البيانات: الأرقام والسلاسل والكائنات والمزيد

Booleans : القيم المنطقية

Booleans can only have two values: true or false.

يمكن أن يكون Booleans قيمتين فقط : صواب أو خطأ

مثال

```
var x = 5;

var y = 5;

var z = 6;

document.getElementById("demo").innerHTML = (x === y) +
"<br>" + (x === z);
```

ففي حالة أن x يساوي y فسوف يقوم المتصفح بطباعة القيمة true

وفي حالة أن x لا يساوي z فسوف يقوم المتصفح بطباعة القيمة false

مثال الزيرو

```
/*
    Boolean: True, False
*/

/*
    Check If The Product Has Discount Or No.
    true = Yes Has Discount
    false = No It Has No Discount
*/

var hasDiscount = false; // False Mean Has No Discount

if (hasDiscount === true) {

    var mainPrice = 350; // If Has Discount

} else {

    var mainPrice = 450; // If Has No Discount

}

document.getElementById("test").innerHTML = mainPrice;
```

Arrays: المصفوفات

المصفوفات arrays تكتب بأقواس مربعة [ ]

مثال / `var cars = ["Saab", "Volvo", "BMW"];`

يتم فصل عناصر المصفوفة بفواصل commas

ترتب عناصر arrays من الصفر بحيث يمكن اختيار عنصر واحد فقط منهم
مثال/

```
document.getElementById("demo").innerHTML = cars[2]
```

في هذا المثال نلاحظ أن المتصفح سيقوم بطباعة العنصر BMW لأننا قمنا بتحديده عن طريق اسم المتغير cars[2]

مثال الزيرو

```
    Array: ["facebook.com", "youtube.com", "google.com"]
*/

var socialWebsites = ["facebook.com", "youtube.com", "google.com"];

document.getElementById("test").innerHTML = socialWebsites[2];
```

Objects :

تتم كتابة كائنات جافا سكريبت باستخدام أقواس معقوفة curly braces **{}**

تتم كتابة خصائص الكائن مثل name:value ، مفصولة بفواصل

مثال/

```
    Object: {firstName: "Osama", lastName: "Mohamed"}
*/

var myInfo = {firstName: "Osama", lastName: "Mohamed"};

document.getElementById("test").innerHTML = myInfo.lastName;
```

Strings

هو عبارة عن text
ويمكن كتابته ب double quotes " "

مثال / var carName = " Volvo XC60 ";

ويمكن كتابته ب single quotes ' '

مثال / var carName = ' Volvo XC60 ';

ويمكن استخدام single quotes **داخل** double quotes

```
var answer = "He is called 'Johnny' ";
```

أو العكس

استخدام double quotes **داخل** single quotes

```
var answer = 'He is called "Johnny"  ';
```

أو استخدام double quotes مرتين مع استخدام السلاش معهم

```
    String: "JavaScript"
*/

var myName = "Osama Mohamed \"Zero\"";

document.getElementById("test").innerHTML = myName;
```

Numbers

الأرقام وتتم كتابتها مباشرةً بدون double quotes " "

```
var Myage = 34;
```

الفرق بين كتابة الأرقام بدون double quotes و مع كتابتها ب double quotes

إذا كتبنا الأرقام بـdouble quotes فإنها تكون string نص وليست أرقام

```
var Myage = "34";
```

فعند القيام بأي عمليه حسابيه فلن يقوم بها لإنه لم يعد رقم بل نص

```
var myAge = "32" + 10; // "32" + "10" = 3110
```

Undefined

وهو متغير بدون قيمة

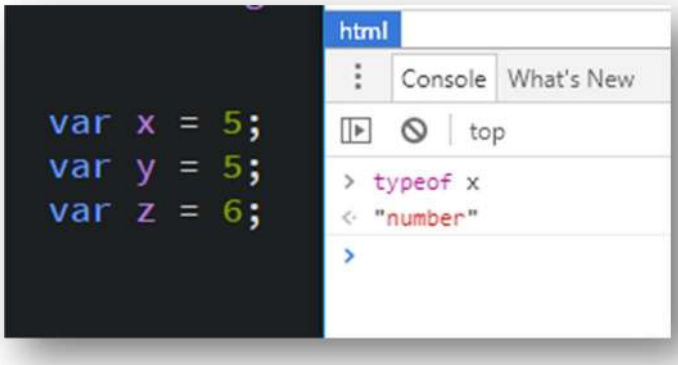
```
var car;
```

القيمة غير معروّفة ، النوع غير معروف

Typeof

إذا كتبنا في console [typsof + Variable Name]

فإنه سوف يخرج لنا نوع البيانات الخاصه بالجافا سكربت



+ : Concatenation

هو ربط الحروف جمباً إلى جمب

```
var myAge = "32" + "10"; // "3210"
```

فهو يقوم بربط الحروف وليس اجراء العملية الحسابية فيكون الناتج هو [3210]

فعلامة الجمع عندما تدخل مع strings فإنها تقوم بربط الحروف concatenation

ملاحظة لا يصح جمع [number + strings]

```
var myAge = 5 + 4 + "Osama؛"
```

النتيجة / 9Osama

نلاحظ / عند جمع الأرقام مع strings وتكون الأرقام في البداية فإن المتصفح يجمعها ويربط معها strings

```
؛var myAge = "Osama" + ٥ + ٤
```

النتيجة / Osama54

نلاحظ / أما عند جمع الأرقام مع strings وتكون strings في البداية فإن المتصفح يربط strings مع الأرقام دون جمع الأرقام

```
var myAge = "Osama" + 4 + 5; // "Osama" + "4" + "5"
```

مثال أخر

```
var myAge = 10 + 5 + "Osama" + 5 + 4؛
```

النتيجة / **15Osama54**

نلاحظ / أن المتصفح قم بجمع الأرقام لأنهم في البداية ثم ربط معهم strings وربط باقي الأرقام التي بعد string

أما إذا تم وضع الأرقام بين أقواس فسوف يتم جمع الأرقام

```
var myAge = 10 + 5 + "Osama" + (5 + 4);
```

النتيجة / **15Osama9**

نقوم بإضافة double quotes " " عندما نريد عمل مسافة

```
document.getElementById("test").innerHTML = "My Name Is:" + " " + myName;
```

My Name Is: Osama

وممكن فقط نقوم بعمل مسافة بإبعاد double quotes

```
"My Name Is: " + myName;
```

```
document.getElementById("test").innerHTML =
    "My Name Is: " + myName + " And My Age Is: " + myAge + " My Country Is: " + myCountry;
```

My Name Is: Osama And My Age Is: 32 My Country Is: Egypt

إذا قمنا بإضافة
 فإنهم سوف يكون كل عنصر في سطر منفصل

```
var myName = "Osama",
    myAge = 32,
    myCountry = "Egypt";
document.getElementById("test").innerHTML =
    "My Name Is: " + myName + "<br>" +
    "My Age Is: " + myAge + "<br>" +
    "My Country Is: " + myCountry;
```

My Name Is: Osama
My Age Is: 32
My Country Is: Egypt

```
"My Name Is: " + "Osama" + "<br>" +
"My Age Is: " + "32" + "<br>" +
"My Country Is: " + "Egypt";
```

فنلاحظ أن النتيجة هي هي ولكن بدلاً من وضع اسم المتغير قمنا بوضع قيمة المتغير مباشرة.

ولكننا نستخدم اسم المتغير لأنه يتعامل مع **databas** مع قاعدة البيانات فإنه عندما يتم تغييره فسوف تتغير نتائجة على صفحة html عكس ما إن تمت كتابه قيمه مباشرة.

ومن الممكن عمل تنسيق للنصوص داخل الجافا سكربت

```
"<span style=\"color:red\">My Name Is</span>: <span style='color:blue'>" + myName + "</span><br>" +
"My Age Is: " + myAge + "<br>" +
"My Country Is: " + myCountry;
```

My Name Is: Osama
My Age Is: 32
My Country Is: Egypt

Output

وهي أخراج الجافا اسكربت أو طريقة اظهار أوامر الجافا سكربت

تستطيع جافا سكريبت "عرض" البيانات بطرق مختلفة

١ : باستخدام.innerHTML

٢ : باستخدام.() document.write

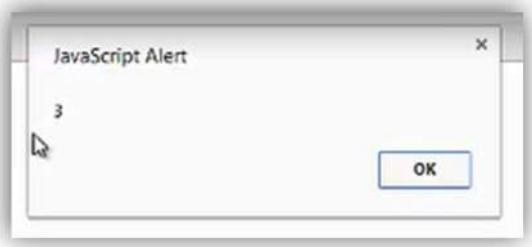
٣ : باستخدام.() window.alert

٤ : باستخدام .().console.log

Alert

هو صندوق تنبيه لعرض البيانات

```
var x = 1,
    y = 2;
alert(x + y);
```

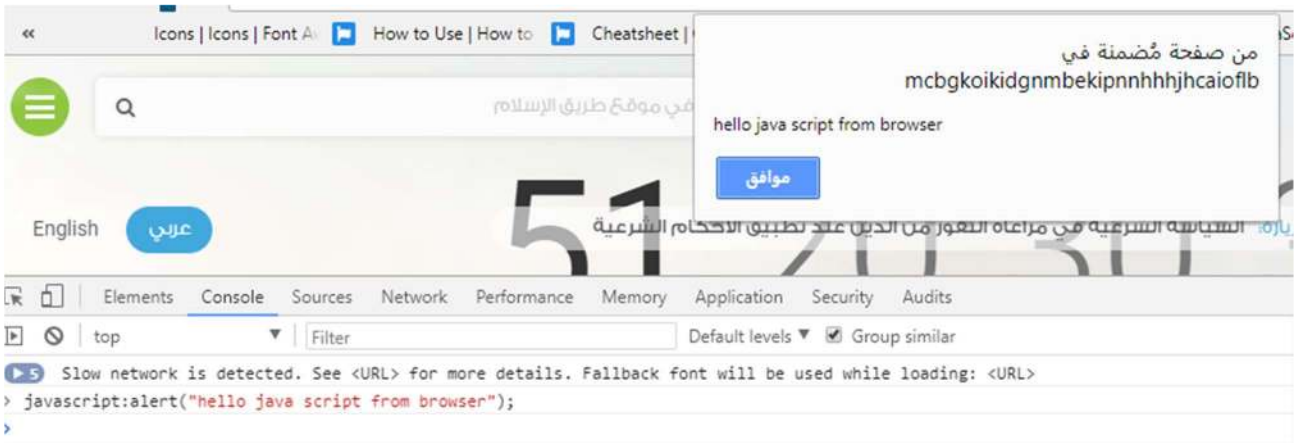


فهو عبارة عن صندوق يظهر فيه أو يقوم بطباعة مخرجات أوامر الجافا سكربت بداخله

ويمكن كتابت الأمر مباشرة في المتصفح

JavaScript:alert("hello java script from browser")؛

ويمكن كتابتها في concoel أيضاً



document.write

وتستخدم لإغراض تجريبية

ولا تستخدم في العمل الفعلي " Real Work "

باستخدام document.write () بعد تحميل مستند HTML بالكامل ، سيتم حذف كل HTML الموجود

مثال/

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<button type="button" onclick="document.write(5 + 6)">Try
it</button>

</body>
</html>
```

فيجب استخدام الطريقة document.write () للاختبار فقط

innerHTML

للوصول إلى عنصر HTML ، يمكن لجافا سكريبت استخدام طريقة
document.getElementById (id)

تحدد السمة id عنصر HTML تحدد الخاصية innerHTML محتوىHTML

فيتم عرض المخرجات داخل صفحة Html

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

<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>


<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
</body>
</html>
```

console.log()

وحدة التحكم

لأغراض تصحيح الأخطاء ، يمكنك استخدام الأسلوب () console.log لعرض البيانات.

كترك رسالة لما فيها للمطور

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

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

فإذا قمت بفتح **console** داخل المتصفح فسوف تجد حاصل جمع ٥+٦

```
/*global console, alert*/

var myName = "Osama",

    myAge = 32,

    myDiv = document.getElementById("test");

// alert("Hello My Name Is " + myName + " And My Age Is " + myAge);

// document.write("Hello My Name Is " + myName + " And My Age Is " + myAge);

// myDiv.innerHTML = "Hello My Name Is " + myName + " And My Age Is " + myAge;

// console.log("Hello My Name Is " + myName + " And My Age Is " + myAge);
```

Operators

علامات الجمع أو العمليات الحسابية

الجمع + Addition

مثال

```
var x = 5;
var y = 2;
var z = x + y; // z = 5 + 2 = 7
document.getElementById("test").innerHTML = z;
```

الطرح - subtraction

مثال

```
var x = 5;
var y = 2;
var z = x - y; // z = 5 - 2 = 3
document.getElementById("test").innerHTML = z;
```

إذا قمنا بعمل طرح لـstaring من number فسوف يخرج لنا هذه الرسالة NAN
ومعناها هذا ليس رقم [not a number]

```
var x = "osama";
var y = 2;
var z = x – y;
```

القسمة / Division

```
var x = 100;
var y = 2;
var z = x / y; // z = 100 / 2 = 50
document.getElementById("test").innerHTML = z;
```

الضرب * multiplication

```
var x = 5;
var y = 2;
var z = x * y; // z = 5 * 2 = 10
document.getElementById("test").innerHTML = z;
```

المتبقي % modulus (remainder)

وهي تعمل على تقريب العمليات الحسابية أو المتبقي

```
var z = 21 % 2;
```

سيكون الناتج ١

لأن ال٢ تقبل القسمة على ٢٠ ويتبقى ١

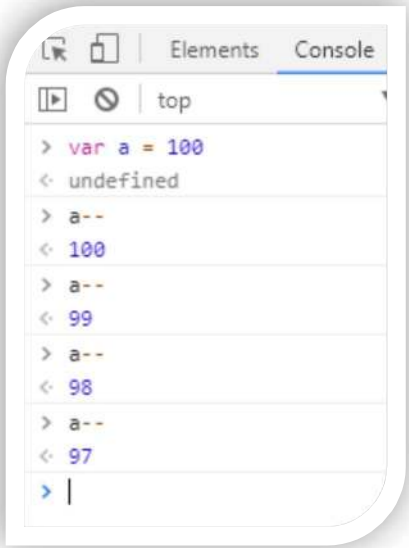
أما إن كانت ٢٠%٢ فستكون النتيجة ٠

Increment ++ زيادة الراتب

وهي تستخدم كعداد زيارة للمواقع

Decrement -- انقاص

وممكن استخدامها في مواقع التجارة في حالة الشراء يقل عدد المنتجات حتى يصل إلى الصفر فتظهر رسالة بانتهاء المنتج



If...Else

يتم استخدام العبارات الشرطية لتنفيذ إجراءات مختلفة بناءً على ظروف مختلفة.
في كثير من الأحيان عند كتابة التعليمات البرمجية ، تحتاج إلى تنفيذ إجراءات مختلفة لقرارات مختلفة.
يمكنك استخدام العبارات الشرطية في التعليمات البرمجية للقيام بذلك.

(>) أصغر من (<) أكبر من

If

استخدم العبارة if لتحديد كتلة من تعليمات JavaScript البرمجية ليتم تنفيذها إذا كان الشرط صحيحًا.

else

استخدم عبارة else لتحديد كتلة من التعليمات البرمجية ليتم تنفيذها إذا كان الشرط غير صحيح.

```
var ticketPrice = 8000; // Ticket Price Variable

if (ticketPrice < 1500) { // If The Ticket Cheaper Than 1500

    console.log("Yes Its Cheap"); // Display Message That Its Cheap

} else {

    console.log("No Its Expensive"); // Display Message That Its Expensive

}
```

شرح المثال/

اظهر في ال console عبارة " yes its cheap " إذا كان ticketprice أقل من ١٥٠٠

Else غير ذلك اظهر عبارة " no its expensive ".

else if

استخدم العبارة " **else if** " لتحديد شرط جديد إذا كان الشرط الأول خاطئًا

```
var ticketPrice = 800; // Ticket Price Variable

if (ticketPrice < 1500) { // If The Ticket Cheaper Than 1500

    console.log("Yes Its Cheap"); // Display Message That Its Cheap

} else if (ticketPrice == 2000) { // If The Ticket Price Is 2000

    console.log("Yes Its Good Price"); // Display Message That Its Good

} else {

    console.log("No Its Expensive"); // Display Message That Its Expensive

}
```

مثال أخر

```
var yourAge = prompt("Whats Your Age?");

if (yourAge < 18) {

    document.getElementById('test').innerHTML =

        "Sorry Your Age Is " + yourAge + " You Are Not Allowed Here";
} else {

    document.getElementById('test').innerHTML =

        "Hello Your Age Is " + yourAge + " You Are Welcome Here";

}
```

وفي هذا المثال نقوم بعمل prompt لتحديد عمر الزائر
فإن كان أقل من ١٨ سنة لا تظهر له العبارة sorry

وإن كان ١٨ فأكثر تظهر له عبارة hello

تسخدم()clear لحذفما تم كتابته في ال consel

Assignment Operators =

يعين قيمة للمتغير

Var myPrice = 50

```
if (myPrice = 60) {
```

```
if (myPrice = 10009090909) {
```

```
var myPrice = 50;

if (myPrice = 50) {

    alert("Good");

} else {

    alert("Bad");

}
```

نلاحظ : هنا أننا مهما قمنا بإضافة أرقام فتظر رسالة ال alert دائما دون تغيير
وهذا يعود إلى **<u>أن علامة (=) الواحدة لا تصلح للمقارنات</u>**

Comparison Operator (==)

وهو يقوم بعمل مقارنات من حيث [value] فقط

```
/*global console, alert*/

var myprice = 50;

if (myprice == 100) {

    alert("Good");

} else {

    alert("Bad");

}
```

فنلاحظ هنا : أن ال value مختلف لهذا ظهرت رسالة ال bad

Identical Operator (===)

وهو يقوم بعمل مقارنات من حيث [data type + value]

```
/*global console, alert*/

var myprice = "50";//Data Type <= string

if (myprice === 50){//Data Type <=
Nummer


    alert("Good");

} else {

    alert("Bad");

}
```

فنلاحظ : أن نوع ال data type == = string والأخرى numper

وال value واحده 50 فظهر الإختلاف

المثال

```
var myprice = "50";   //Data Type <= string

if (myprice === 50){//Data Type <= Nummer

alert("Good")}

 else { alert("Bad");}
```

```
/*
    ( = ) Assignment Operator
    ( == ) Comparison Operator Comapre Value
    ( === ) Identity Operator Comapre Data Type + Value
*/
```

Logical Operators

العلامات المنطقية

لا يساوي Not Equal (!)

(!=) ! Not = Equal (Not Equal)

وهو يعني لا يساوي ال value

مثال /

```
var age = "50";

if (age != 50) {

    alert("Good");

} else {

    alert("Bad");

}
```

نلاحظ : أننا نخبره أن ال age لو هو لا يساوي 50 فأخرج لي رسالة Good
فقام المتصفح بإخراج رسالة Bad لأنه فعلاً يساوي 50

(!==) Not Identical

وهو بمعنى لا يساوي [Data Type + Value]

فإذا تحقق شرط ولم يتحقق الأخر فإن الأمر ينفذ

```
var age = 50;

if (age !== 50) {

    alert("Good");

} else {

    alert("Bad");

}
```

نلاحظ هنا / أن age تساوي قيمة المتغير ومشتركه معاه في نوعة فلهذا أخرج المتصفح الرسالة Bad

لكن عندما اختلفت في data type اختلف الأمر

And (&&) و

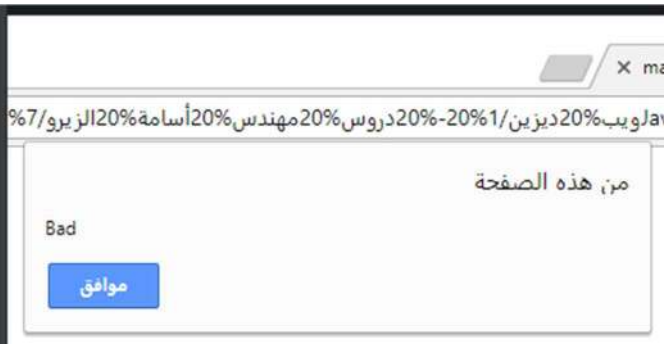
وتستخدم في حالة دمج الشروط بمعنى [**أنه يجب أن تتحقق جميع الشروط لينفذ الأمر اللذي تريده**] وهو يركز على value في حالة استخدام ==

مثال /

نلاحظ / تحقق الشرطين حيث أن الأسم متحقق وال value متحقق

أما في المثال الأتي فنلاحظ أختلاف date type فالعمر في الأولى string وفي الثانية Nambur

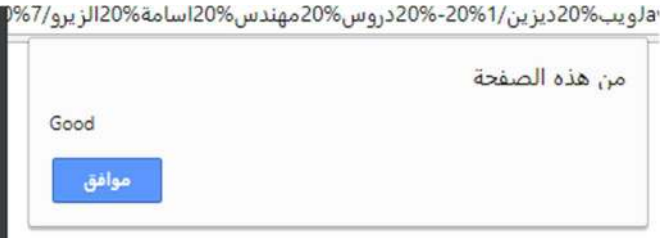
```
var
    name = "Osama",

    age = "50";

if (age === 50 && name === "Osama")
{

    alert("Good");

} else {

    alert("Bad");

}
```



or (||) أو

وتستخدم في حالة التخيير بين الشروط بمعنى **[أنه إذا تحقق شرط من مجموعة هذه الشروط نفذ لي هذا الأمر]**

```
var
    name = "Osama",

    age = "50";

if (age === 50 || name === "Osama")
{

    alert("Good");

} else {

    alert("Bad");

}
```



نلاحظ : في هذا المثال تحقق شرط واختلف شرط ومع هذا فقد نفذ الأمر وأظهر رسالة Good

Function

المهام أو الوظائف

هي مجموعة من الأكواد المصممة لتنفيذ مهمة معينة.

لماذا وظائف؟

يمكنك إعادة استخدام الرمز: حدد الرمز مرة واحدة ، واستخدمه عدة مرات.

يمكنك استخدام نفس الرمز عدة مرات باستخدام وسيطات مختلفة ، لإنتاج نتائج مختلفة.

Function Syntax

بناء الوظيفة

الكلمة المفتاحية أو keyword الخاصة بها في الكود Function.

يتم تعريف function باستخدام الكلمة الدالة ، متبوعة باسم ، متبوعة بأقواس() .

function sayHi؛()

```
function sayHi();
```

Function names : اسم الوظيفة

يمكن أن تحتوي أسماء الدوال على أحرف ، وأرقام ، وشرطات سفلية ، وعلامات دولار (نفس القواعد مثل المتغيرات).

قد تتضمن الأقواس أسماء المعاملات مفصولة بفواصل:

Function name(parameter1, parameter2, parameter3)

يتم وضع الشفرة المطلوب تنفيذها ، بواسطة الدالة ، داخل أقواس معقوفة { } :

Function Invocation

استدعاء الوظيفة أو تنفيذها

يمكن تنفيذ ال الوظيفة باستدعائها بكتابة function Name اسم الوظيفة كما في المثال

```
function sayHi(){

var myName = "Osama";

alert("Hello" + myName + "Form Insid The function");

}

sayHi();
```

ويمكن استدعائها بإضافة button في صفحة html

```
<button onclick="sayHi()">Call The Function</button>
```

```
function sayHi() {

    "use strict";

    var myName = "Osama";

    alert("Hello " + myName + " Form Insid The function");

}
```

Function Return

وظيفة العودة

عندما تصل JavaScript إلى عبارة return ، ستتوقف الدالة عن التنفيذ.
إذا تم استدعاء الدالة من عبارة ، فستقوم جافا سكريبت "بإرجاع" لتنفيذ التعليمات البرمجية بعد العبارة الاستدعاء.
غالبًا ما تقوم الدوال بحساب قيمة الإرجاع. يتم إرجاع "إرجاع" القيمة إلى "المتصل:"

```
var x = myFunction(4, 3);

document.getElementById("demo").innerHTML = x;

function myFunction(a, b) {
    return a + b;
}
```

نلاحظ في المثال طباعة رقم 7 وهذا لأنه عند طباعة x فهو يجد أمر إدخاله على ال myfunction فيدخل عليه فيجد أمر return بجمع parameter داخل الأقواس فيقوم return بإرجاع القيم الأصلية وجمعها.

مثال أخر

```
function myInfo() {

    "use strict";

     var myName = "mamdouh",

         myAge = 39;

    return myAge;

}

var myfunction = myInfo();


document.getElementById("demo").innerHTML = myfunction;
```

ومن الممكن داخ ال return أن نقوم بعمل عملية حسابيه

```
return myAge + 20;
```

وممكن عمل عملية حسابيه ليس لها علاقة بال function

```
return 100 + 20;
```

Function Parameters

معاملات الوظائف

يتم سرد معلمات الدالة داخل الأقواس ( ) في تعريف الدالة.

وسيطات الدالة هي القيم التي تتلقاها الدالة عند استدعائها.
داخل الدالة ، تتصرف الوسائط (المعلمات) كمتغيرات محلية.

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

الوظيفة هي نفسها مثل إجراء أو روتين ، في لغات البرمجة الأخرى.

```
function sayHello(theName) {

    "use strict";

    return "Hello " + theName + " How Are You Today";

}

document.getElementById("demo").innerHTML = sayHello("Hassan");
```

Parameter

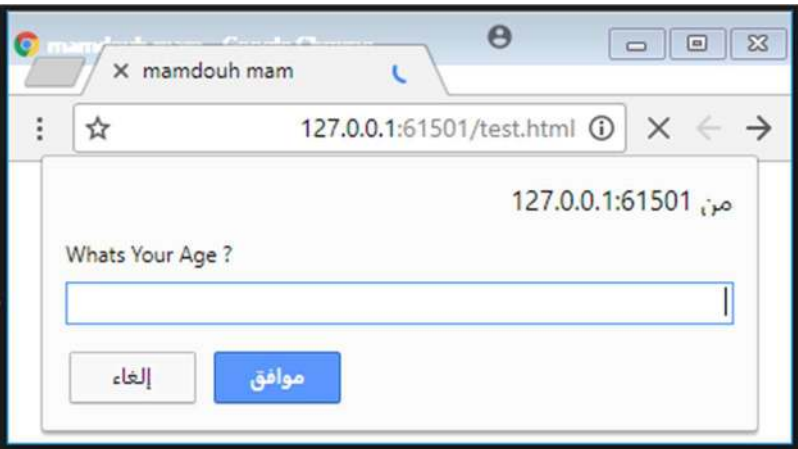
نلاحظ في هذا المثال / أنه تم تحديد parameter مسبقاً لsayHello وهي " Hassan" فعند طباعتها يدخل على الfunction يجد return يخبره بأن يرجع parameter الأصلي له وليس theName بالإضافة لطباعة عبارة الترحي

مثال أخر/

```
function makeDiscount(price) {

    "use strict";

    var discount = 200;


    return price - discount; // 600 - 200 = 400

}

document.getElementById("test").innerHTML = makeDiscount(600);
```

مثال أخر لحساب العمر بالأيام

```
function calcDays(age) {

    "use strict";

    var year = 365;


    return age * year; //

}

var dynamicAge = prompt("Whats Your Age ?")

document.getElementById("test").innerHTML = calcDays(dynamicAge);
```



الترتيب في الجافا سكربت غير معتمد على بعضه
بمعنى ممكن أكتب الجافا اسكربت وبعدين استدعيها

مثال /

```
myFunc();

function myFunc() {

    "use strict";

    alert("My Function");
}
```

Self invoke

وهي أن يعمل ال function بمجرد فتح الصفحة بدون استدعاء

ونقوم بإضافه قوس قبل ال function وقوس بعدها ثم نضيف قوسين ال function وعلامة ; لغلق ال function

```
(function sayWelcome() {

    "use strict";

    alert("Welcome To My Website");

})();
```

يفيد ال self invoke في مواقع البيع والشراء عند وجود سلعة ما فيعمل على تحويل السعر حسب دولة الزبون المستخدم للموقع

مثال أخر

```
<div id="pricee">100</div>
```

```
function convertUsdToRiyal() {

    "use strict";

    var amount = document.getElementById("price").innerHTML; // 100

    alert(amount * 3.75);

}

convertUsdToRiyal();
```

قمنا بعمل div في ال html ثم في JavaScript قمنا بعمل function سوف يأخذ ١٠٠ من قاعدة البيانات ويضربها في ٣,٧٥وقمنا باستدعاء للfunction

مثال أخر

سنقوم بعمل حقل أدخال input داخل صفحة ال Html

```
<body>
    <input type="text" id="dollar">
    <button onclick="conve()">calculate</button>
    <div id="message"></div>

 <script src="testjava.js"></script>
</body>
```

```
/*global document*/

function conve() {

    "use strict";

    var amount = document.getElementById("dollar").value,

        result = amount * 3.75, // 100 * 3.75 = 375

        message = document.getElementById("message");


message.innerHTML = amount + " Dollar Is Worth " + result + " " + "Riyal";


}
```

إذن ال amount هي ال valuo التي سوف نقوم بإدخالها في حقل الإخال

```
var amount = document.getElementById("dollar").value,
```

والresult هي ال value التي سوف نقوم بإدخالها مضروبة في ٣,٧٥

```
result = amount * 3.75;
```

```
message = document.getElementById("message");
```

سوف if في المثال

```
if (amount === "") {message.innerHTML = "أدخل رقم صحيح";

} else {

message.innerHTML = amount + " Dollar Is Worth " + result + " " + "Riyal";
}
```

```
if (amount === "")
```
نخبره هنا بأنه if لو كانت value فارغة empty

وقام بالضغط على الزر اخرج له رساله " أدخل رقم صحيح "
else غير ذلك أخرج الرسالة الأخرى.

```
} else if (isNaN(amount)) {

 message.innerHTML = "ما قمت بإدخاله ليس رقم";

} else {
```

هنا نضيف شرط أخر

```
} else if (isNaN(amount)) {
```
لو كانت amount ليس رقم يعني ال valuo المكتوبة في حقل الإدخال ليست رقم أخرج هذه الرسالة

ونضيف لل () else if عبارة isNaN وهي تعني ليس رقم.

```
else if (amount === "0") {message.innerHTML = "لا تقم بإدخال الصفر";}
```

في هذا المثال أردنا أن لا يتم استخدام صفر في حقول الإدخال

فائدة / نلاحظ أننا كتبنا الصفر ك string وليس number وهذا لأن حقل الإدخال text

```
else if (amount < 0) {message.innerHTML = "أدخل رقم أكبر من الصفر";}
```

نلاحظ : هنا أننا قمنا بإضافة شرط أخر وهو أن يكون الرقم أكبر من الصفر وليبس بالسالب.

المثال كامل /

```
/*global document*/
function conve() {
    "use strict";
    var amount = document.getElementById("dollar").value,
        result = amount * 3.75, // 100 * 3.75 = 375
        message = document.getElementById("message");
   if (amount === "") {message.innerHTML = "أدخل رقم صحيح";}
    else if (isNaN(amount)) {message.innerHTML = "ما قمت بإدخاله ليس رقم";}
    else if (amount === "0") {message.innerHTML = "لا تقم بإدخال الصفر";}
    else if (amount < 0) {message.innerHTML = "أدخل رقم أكبر من الصفر";}
    else {message.innerHTML = amount + " Dollar Is Worth " + result + " " + "Riyal";}
}
```

JavaScript Switch

يستخدم بيان التبديل لتنفيذ إجراءات مختلفة بناءً على ظروف مختلفة.
استخدم العبارة switch لتحديد أحد كتل عديدة من التعليمات البرمجية ليتم تنفيذها.
يحتوي المحول على مجموعة أو أكثر من قوالب الحالة وعلامة اختيارية اختيارية.

يعني هو شبيه بالإختيار من متعدد أو وضع أجابه صحيح وفي حالة وضع اجابه خطأ تظهر رساله ما

كيف يعمل؟

- يتم تقييم تعبير التبديل مرة واحدة

- تتم مقارنة قيمة التعبير مع قيم كل حالة.
- في حالة وجود تطابق ، يتم تنفيذ كتلة التعليمات البرمجية المقترنة.

مثال /

```
/*global document, prompt, alert*/

var season = prompt("What The Best Season For You?");

switch (season) {

case "Winter":
    alert("Winter Is Too Cold");
    break;
case "Summer":
    alert("Summer Is Too Cold");
    break;
case "Autumn":
    alert("Autumn Is Very Good");
    break;
case "Spring":
    alert("Spring Is Amazing");
    break;
default:
    alert("You Didnt Enter A Season Name");
    break;
}
```

```
/*global document, prompt, alert*/

var season = prompt("What The Best Season For You?");

switch (season) {

case "Winter":
    alert("Winter Is Too Cold");
    break;
case "Summer":
    alert("Summer Is Too Cold");
    break;
case "Autumn":
    alert("Autumn Is Very Good");
    break;
case "Spring":
    alert("Spring Is Amazing");
    break;
default:
    alert("You Didnt Enter A Season Name");
    break;
}
```

JavaScript Scope

نطاق جافا سكريبت

يحدد النطاق إمكانية الوصول (الرؤية) للمتغيرات.

يعني نطاق عمل الجافا اسكربت وتنفيذ الأوامر

هناك نوعان من نطاق الجافا سكربت :

نطاق داخلي.. Local scope

المتغيرات المحددة داخل وظيفة لا يمكن الوصول إليها

من خارج الوظيفة.

```
/*global console*/

function testparent() {
    "use strict";
   var x = 5,
    calc = x + 3;
    console.log(calc);
    }
testparent();
```

مثال /

ال console.log(calc) لا يمكن أن تعمل إن كتبناها خارج ال function لأنها Local نطاق عملها محلي أي داخل ال function فقط ولا يمكن استدعائها من الخارج.

نطاق عام Global scope

ويمكن استدعائه من أي مكان حتى داخل الfunction

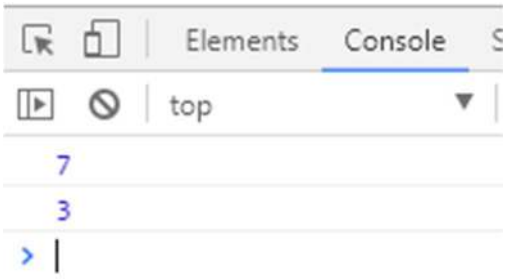
نلاحظ /

في هذا المثال أن ال var متغير global يمكن استدعائه من أي مكان حتى من داخل ال function وبعد كتابة ال Function كذلك.

```
var x = 2;
function testparent() {
    "use strict";
    function testChild() {
        var calc = x + 3;
        console.log(calc);
}
testChild();
}
testparent();
console.log(x + 1);
```

مثال

```
/*global console*/
var x = 1;
function testparent() {
    "use strict";
    var x = 5;
    function testchild() {
        var calc = x + 2;
        console.log(calc);
    }
    return testchild();
}
testparent();
console.log(x + 2);
```



نلاحظ أننا لا نستطيع الدخول للـ function الأبن مباشرة إلا بعد أن استخدمنا أمر return لإستدعاء function الأبن داخل الأب لأنه Local أما الـ function الأب فهو Global

فأي function داخل function يعتبر Local أي فرعي لا تستطيع الدخول إليه مباشرةً والـ function الأب له يعتبر Global

JavaScript Events

الحدث

عند استخدام JavaScript في صفحات HTML ، يمكن لـ" JavaScript التفاعل" في هذه الأحداث.
يمكن أن يكون حدث HTML شيئًا يفعله المتصفح ، أو شيء يفعله المستخدم.

وهي تساعدك في حدوث تغير في صفحة Html كا عمل دونلود لحين تحميل

Window.onload

معناها بعد تحميل الصفحة طبق لي هذا الحدث

فسوف يقوم بتحميل كل عناصر الصفحة ثم بعد ذلك سوف يقوم بتطبيق الكلام أو الحدث اللذي كتبته له

```
<head>
     <meta charset="utf-8">
    <link rel="stylesheet" href="test1.css">
     <title>Mamdouh</title>
    <script>
        window.onload = function () {
            document.getElementById("demo").innerHTML = "Text From JavaScript"
        }
    </script>
</head>
<body>
    <div id="demo"></div>
</body>
```

Text From JavaScript

Window.onclick

ومعناها طبق لي هذا الحدث عندما أقوم بالضغط على الصفحة ضغطة واحده فقط

```
<script>
    window.onclick = function () {
        document.getElementById("demo").innerHTML = "Text From JavaScript"
    }
</script>
```

Window.ondblclick

ومعناها طبق لي هذا الحدث عندما أقوم بالضغط على الصفحة ضغتتين أي مرتيين

```
<script>
    window.ondblclick = function () {
        document.getElementById("demo").innerHTML = "Text From JavaScript"
    }
</script>
```

مثال لإستخدام onclick & ondblclick مع button

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="test1.css">
    <title>Mamdouh</title>
    <script>
        function changeMe() {
            document.getElementById("demo").innerHTML = "Text From JavaScript"
        }
    </script>
</head>
<body>
    <button onclick="changeMe()">Vhange Me</button>
    <div id="demo"></div>
</body>
</html>
```

فهنا قمنا بإنشاء زرار وجعلنا بالضغط عليه مرة واحده يتحقق الحدث وEvents

وممكن تغير onclick لي ondblclick

```
<button ondblclick="changeMe()">Vhange Me</button>
```

ومثال أخر لإستغلال button لتحقيق ٢ حدث فيه

```
    <script>
        function changeMe() {
            document.getElementById("demo").innerHTML = "قمت بضغطه واحده"
        }
         function change2() {
            document.getElementById("demo").innerHTML = "قمت بعمل ضغطتين"
        }
    </script>
</head>
<body>
    <button onclick="changeMe()" ondblclick="change2()">Vhange Me</button>
    <div id="demo"></div>
</body>
```

Onkeydown

وهي بمجرد لمس الزرار والمرور عليه يحدث الevents

Onkeypress

وهي بعد الضغط على الزرار يحدث ال events

Onkeyup

وهي بعد الضغط على الزرار ورفع اليد عنه يحدث ال events

مثال عن طريقتين لكتابة bottun

الطريقة الأولى : كتابة function في html

```
<body>
    <input id="input" type="text">
    <button onclick="calcUsd()">Calculate</button>
    <div id="div"></div>
    <script src="Events1.js"></script>
</body>
```

```
/*global console*/

var myInput = document.getElementById("input"),

    myDiv = document.getElementById("div");

function calcUsd() {

    "use strict";

    myDiv.innerHTML = myInput.value * 3.75;
}
```

37.5

نلاحظ / في هذا المثال نكتب اسم ال function في html

<button onclick="**calcUsd()**">Calculate</button>

الطريقة الثانية : الكتابة مباشرة في ملف JavaScript

```
<body>
    <input id="input" type="text">
    <button id="button">Calculate</button>
    <div id="div"></div>
    <script src="Events1.js"></script>
</body>
```

```
/*global console*/

var myInput = document.getElementById("input"),

    myDiv = document.getElementById("div"),

    myButton = document.getElementById("button");

myButton.onclick = function() {

    "use strict";

    myDiv.innerHTML = myInput.value * 3.75;
}
```

نلاحظ / قمنا بإعطاء id فقط لل button وكتبنا الأوامر مباشرة في ملف الجافا سكربت

ولم نكتب اسم لل function لأننا لم نعد بحاجة إليه

وأخبرناه كما في المثال الأول يطبع ال value التي نكتبها مضروبة في 3.75 ويخرج لنا الناتج في div

بناء الحدث : events Syntax

Element.Event = function () { code }

```
myButton.onclick = function() {

    "use strict";

    myDiv.innerHTML = myInput.value * 3.75;
}
```

وتعد هذه الطريقة أفضل لأنها تجمع كل ما يخص الجافا اسكربت في ملف منفصل لا علاقة لملف html بها

مثال على onkeydown & onkeyup & onkeypress

```
<body>
    <input id="input" type="text">
    <div id="div"></div>
    <script src="Events1.js"></script>
</body>
```

```
/*global console*/

var myInput = document.getElementById("input"),

    myDiv = document.getElementById("div");

myInput.onkeyup = function() {

    "use strict";

    myDiv.innerHTML = myInput.value * 3;
}
```

Onmouseover

وهي مثل hover في Css

مثال

```
<body>
    <div id="div">Hello</div>
    <script src="Events1.js"></script>
</body>
```

```
var myDiv = document.getElementById("div");

myDiv.onmouseover = function () {

    "use strict";

    myDiv.innerHTML = "You Hovered On Me";
};
```

Onmouseout

وهي بعد مرور الماوس عليها والإبتعاد عنها يحدث الحدث events

Onchange

```
<body>
    <input id="input" type="text">
    <select id="currency">
        <option value="8">Doller</option>
        <option value="2">Riyal</option>
        <option value="4">Yin</option>
    </select>
    <div id="div"></div>
    <script src="Events1.js"></script>
</body>
```

```
var
    myInput = document.getElementById("input"),

    myDiv = document.getElementById("div"),

    myCurrency = document.getElementById("currency");

myCurrency.onchange = function () {

    "use strict";

    myDiv.innerHTML = myInput.value * myCurrency.value;
};
```

80

تم استخدام select مع ال onchange لأظهار المثال

```
myDiv.innerHTML = myInput.value * myCurrency.value;
```

ومعنا هذا السطر اطبع لي داخل ال div ال value التي أقوم بإدخالها مضروبه في ال value الموجودة مسبقاً myCurrency

انتهى الجزء الأول
الجزء الثاني بداية من Array الدرس رقم ٢٥