

كتابة عنوان الصفحة وتعيين اللون والحجم له

```
document.write("<h1>page title</h1>");
```

```
document.querySelector("h1").style.color="blue"
```

```
document.querySelector("h1").style.fontSize="80px"
```

كتابة تعليق قبل بدء الصفحة

```
window.alert("هذا الموقع ملغم بالفيروسات");  
طباعة رسالة داخل الكونصول  
console.log("Hello world");
```

هى عبارته عن واجهة برمجية تقدم لك ماتيريد فعله من خلال لغة API:
جافا مثل التعامل مع الكونصول

ماهو Ecma Script:

هى منظمة عالمية مسؤولة عن معايير المعلومات

```
// الكود التالى يوضح تنفيذ الامر بعد تحميل الصفحة وهكذا يمكنك وضع كود  
// السكربت فى اى مكان فى الصفحة سواء قبل قفلت البودى او قبل قفلت الهيد  
window.onload=function() {  
    document.querySelector("h1").style.color="blue";  
};  
var My_name="osama";  
// تم وضع علامه السى فى المية لتحديد ما يتم تطبيق الخصائص عليه من حيث اللون  
// وغيره  
console.log("hello %cworld","color:red; font-size: 40px");  
console.error("Error");  
console.table(["hossam" , "ahmed" , "mohammed"]);  
console.log('Hello ${My_name}');
```

```

الكود التالي يوضح اعتبار الاسم كامل في سطر واحد
console.log("hossam \
shoukry\
abulfatah");
الكود التالي يوضح وضع كل كلمة في سطر
console.log("hossam\nshoukry\nabulfatah");
الكود التالي يوضح امكانية وضع علامة تنصيص داخل علامة تنصيص اخرى
console.log("hossam elden \"shoukry\"");

```

```

/* concatenation
وظيفةها هي ان تقوم بربط البيانات ببعضها
*/

let a = "we love";
let b = "hossam";
document.write(a + " " + b);

```

```

الكود التالي يوضح تنفيذ الامر بطريقة مختصرة بدلا من وضع
علامة الزائد والتنصيص وغيره
*/
let a = "we love";
let b = "hossam";
let c = "elden";
let d = "shoukry";
الكود التالي يوضح لك امكانية عمل كود اتش تى ام ال بطريقة الجافا سكريبت
let markup=`
<div class="card">
<div class="child">
<h1> Wellcom </h1>
<p> My channel </p>
</div>
</div>
`

console.log(`${a} ${" "} ${b}
${c}${d}`);
document.write(`${a} ${" "} ${b} ${c}${" "} ${d}`);
document.write(markup);

```

يمكنك أيضا وضع متغيرات داخل اتش تي ا مال كالتالي

```
let markup=`  
<div class="card">  
<div class="child">  
<h1> ${a} ${b} </h1>  
<p> ${c} ${" "} ${d} </p>  
</div>  
</div>  
`
```

// جمع متغيرين بطريقة صحيحة

```
let a = 10;  
let b = 20;  
console.log (+a +b);
```

```
let a = 10;  
a = a+20;  
a = a+70;  
a+=100; // this mean that the new a has been added to the old  
a      a=a + 100  
a-=50; // a = a-50  
a /=50; // a = a / 50  
console.log(a);
```

consol.log(1e6) معناها اطبع رقم واحد وجمبه 6 اصفار

consol.log(Number.Max_Safe_Integer); اعلى قيمة امنة

consol.log(Number.value); اعلى قيمة

اذا تم اضافة قيمة الى اعلى قيمة فلن يضيف شيء

Mathods

هى اجراء من الاجراءات التى تنفذ طلب معين.

```
/*
يفهم منك البرنامج انك تعرض الرقم بدون كسور حاليا ويرجع لك الرقم الذى وضعته ولكن بصورة نصية وليست
رقمية
*/

console.log(100..toString());
/*
تتيح لك التحكم فى عدد الارقام المكتوبة بعد العلامة العشرية to fixed
*/
console.log(10.555555.toFixed());
// لاحظ هنا انه يتم وضع رقمين فقط بعد العلامة العشرية
console.log(10.555555.toFixed(2));
```

بترجع لك القيمة نصية سترينج to fixed

شرح (parseInt, parseFloat) :-

بترجع لك القيمة رقمية ولا ترجع قيمة كسرية

```
// NaN عند كتابة نوع سترينج قبل الرقم سوف تعيد لك انه غير رقمى
console.log(parseInt("hossam 100"));
// ترجع لك القيمة الكسرية الرقمية فى المخرجات
console.log(parseFloat("100.500 hossam "));
```

شرح isInteger :-

تقوم باختبار ما اذا كان العدد صحيح ام لا اذا كان العدد مكتوب بطريقة عشرية او نصية
تقوم بارجاع false اذا كان مكتوب بطريقة صحيحة تقوم بارجع true

```
/*
تتيح لك التحكم فى عدد الارقام المكتوبة بعد العلامة العشرية to fixed
*/

console.log(Number.isInteger(100.500 )); //false
console.log(Number.isInteger("hossam")); //false
console.log(Number.isInteger(100)); //true
```

Math round And Math Ceil And Math floor

```
console.log(Math.round(99.2)) // اذا كانت القيمة العشرية تحت النص سوف يلغى القيمة
العشرية ويكتب 99
console.log(Math.round(99.5))// اذا كانت القيمة العشرية على النص او اكثر سوف يلغى
القيمة العشرية ويكتب 100
console.log(Math.ceil(9.2)); // تقريب الرقم لاعلى قيمة حيث ستكون المخرجات هو الرقم
10
console.log(Math.floor(9.9)); // تقريب الرقم لاقل قيمة حيث ستكون المخرجات هي الرقم 9
console.log(Math.min(10 , 20 , 30 , -100)); // تعطى لك اقل قيمة
console.log(Math.max(10 , 20 , 30 , -100)); // تعطى لك اعلى قيمة
console.log(Math.pow(2 , 4 )); // الرقم 2 مرفوع لاس 4
console.log(Math.trunc(99.6)); // تلغى اى قيمة كسرية
```

```
let theName="hosam";
console.log(theName[2]); // الوصول الى الحرف المطلوب access with index
بطريقة الاندكس اى قم بطباعة الحرف رقم 2 وهو حرف الـ s حيث ان بداية العد تكون من الرقم 0
console.log(theName.charAt(2)); // نفس الغرض من الاندكس ولكن قم بعمل التغييرات كما
فى الكود
console.log(theName.length); // طول قيمة المتغير
```

يجب ان تضع فى عين الاعتبار ان المسافة النصية داخل قيمة المتغير تكون محسوبة

لاحظ فى السطر البرمجى التالى عند اضافة فراغ اننه قد تغيرت المخرجات

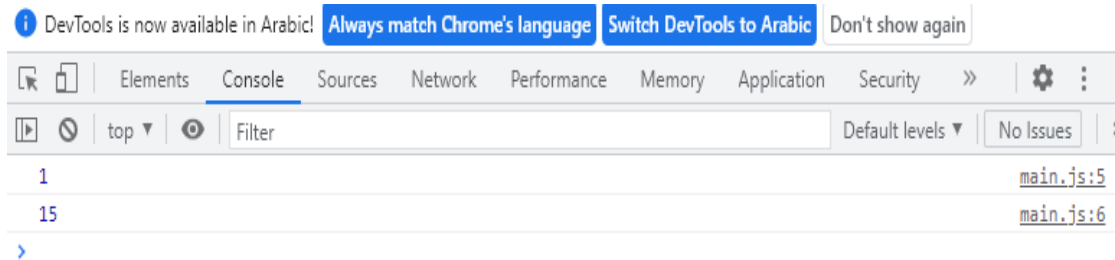
```
let theName=" hosam ";
console.log(theName[2]);
console.log(theName.charAt(2));
console.log(theName.length);

console.log(theName.trim()); // تقوم دالة التريم بازالة المسافات داخل قيمة المتغير
console.log(theName.toUpperCase()); // تحويل القيمة الى حروف كابيتل كبيرة
console.log(theName.toLowerCase()); // تحويل الحروف الى سمول صغيرة
console.log(theName.trim().charAt(3).toUpperCase()); // قم باخلاء الفراغ
داخل المتغير وتحويل رقم الحرف المحدد الى حرف كابيتل

console.log(a.indexOf("Eldin", 8)); // قم بالبحث عن كلمة الدين بدأ من الحرف
الثامن ستكون المخرجات 1- لان بداية الكلمة قد تجاوزت الرقم المحدد
console.log(a.lastIndexOf("Eldin")); // المخرجات تكون رقم 7 لان هذه الدالة تقوم
بالعد بدأ من اخر كلمة
```

Example for index of and last index of:-

```
let a = "Hossam Eldin Shoukry";
console.log(a.indexOf("o"));
console.log(a.lastIndexOf("o"));
```



الاولى كانت المخرجات 1 والثانية 15 لان لاست اندكس تقوم بالعد من اخر كلمة مع الوضع في الاعتبار ان الفراغات تكون محسوبة في عملية العد .

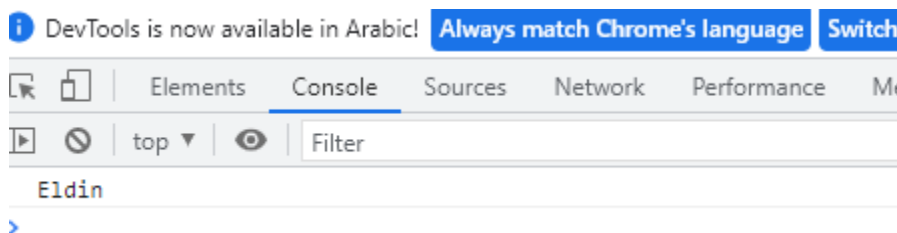
```
let a = "Hossam Eldin Shoukry";
console.log(a.slice(7));
```

وظيفة السلايس انها تقوم بقص جزء من قيمة المتغير على حسب رقم الموضوع في الدالة على سبيل المثال في الدالة الوجوده سيتم قص 6 اجزاء من الاسم حسام مع الوضع في الاعتبار ان الرقم السابع سيكون غير محسوب في عملية القص



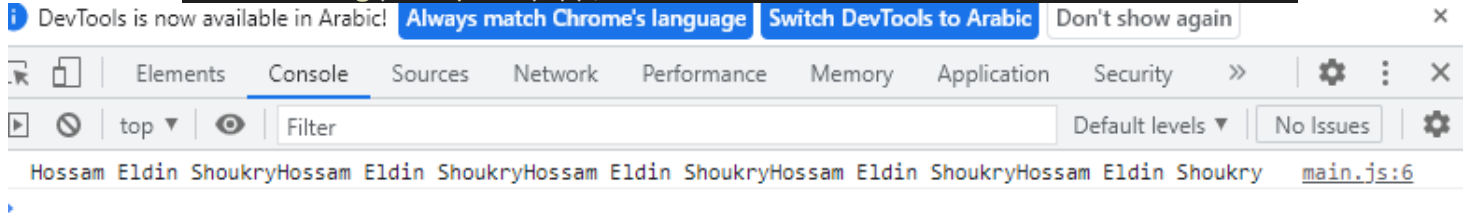
الان نريد اظهرها ر مثلا كلمة الدين بمفردها.

```
let a = "Hossam Eldin Shoukry";
console.log(a.slice(7,12));
```



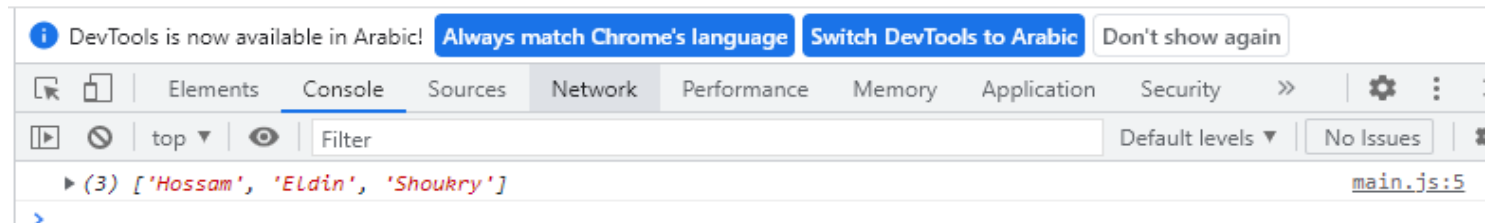
لتكرار المتغير عدة مرات نستعمل دالة repeat كما هو موضح في المثال التالي حيث تم تكرار النص 5 مرات لانه تم وضع الرقم 5 في الدالة:-

```
let a = "Hossam Eldin Shoukry";  
console.log(a.repeat(5));
```



دالة القص split تقوم بقص العنصر حسب ماتريد.

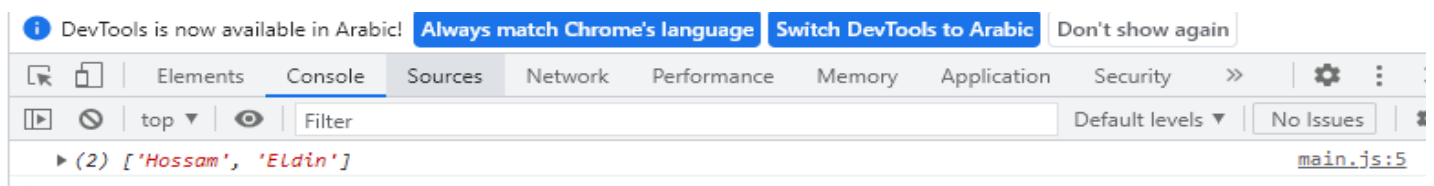
```
let a = "Hossam|Eldin|Shoukry";  
console.log(a.split("|"));
```



في المثال السابق تم قص القيمة من عند العلامة (|).

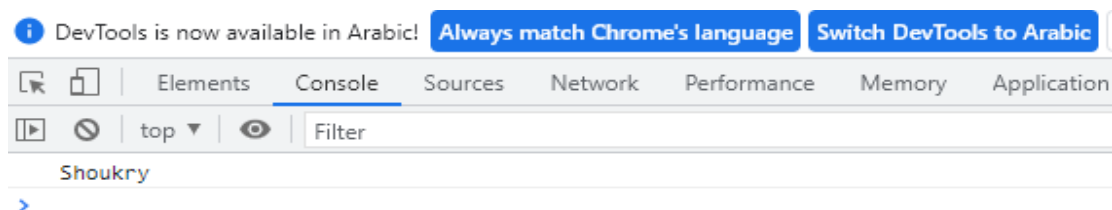
يمكن ايضا تحديد عدد مرات القص كما في المثال التالي :-

```
let a = "Hossam|Eldin|Shoukry";  
console.log(a.split("|", 2));
```



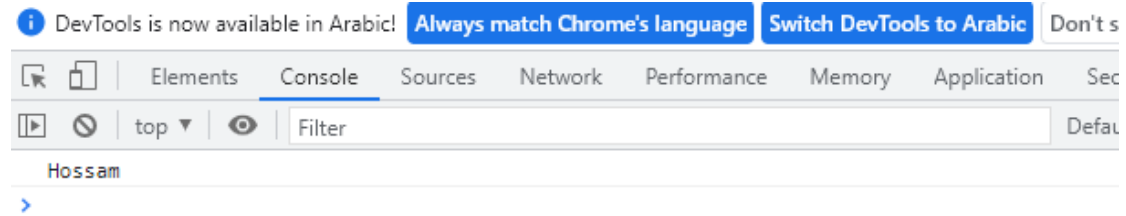
دالة substring تقوم بنفس عمل السلايس ولكن دالة السلايس تعد من اخر رقم اذا تم وضع القيمة بالسالب كما في التالي :-

```
let a = "Hossam Eldin Shoukry";  
console.log(a.slice(-8));
```



مثال على دالة Substring:-

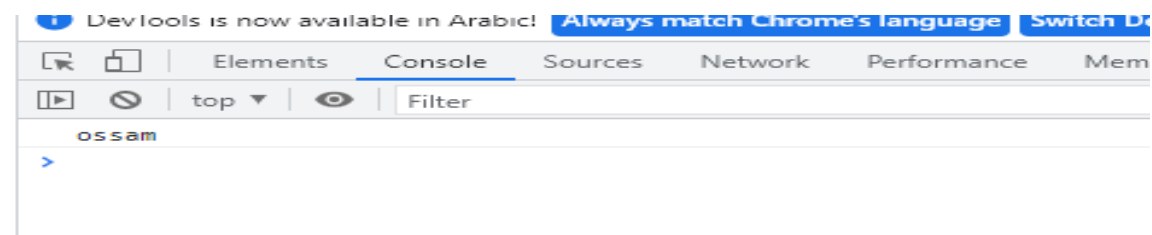
```
let a = "Hossam Eldin Shoukry";  
console.log(a.substring(-8,6));
```



لاحظ هنا في المخرجات ان الدالة بدأت من اول كلمة حتى لو كانت القيمة معطاه بالسالب وليس من اخر كلمة كما في دالة سلايس .

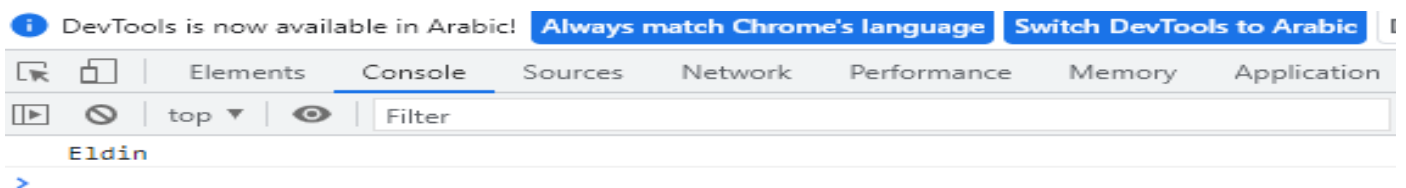
- اذا كانت القيمة المعطاه معكوسة بدأ من الرقم الاكبر كما في المثال التالي لاحظ ان الدالة تقوم باعدة ضبطها تلقائيا لتعطي المخرجات بدأ من الرقم الصغير.

```
- let a = "Hossam Eldin Shoukry";  
- console.log(a.substring(6,1));
```



المثال التالي يوضح لك تحديد جزء من النص بطريقة استخدام الطول length ثم العدد الاكبر تنقص منه واحد لانه يشمل الناهية (not including end) .

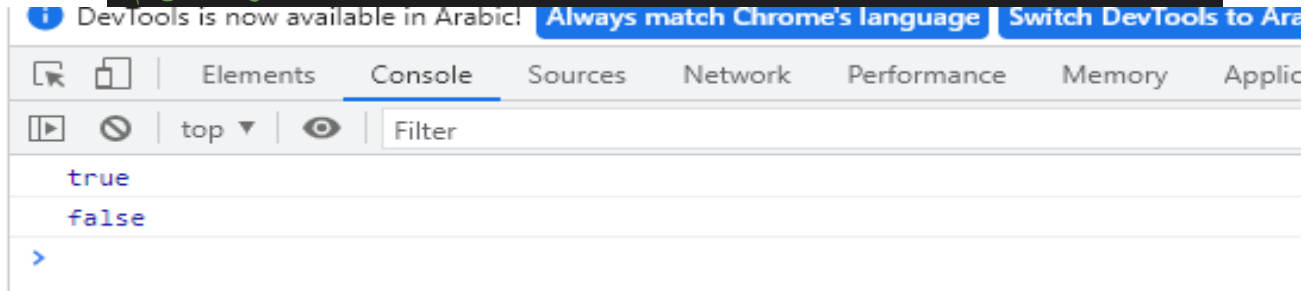
```
let a = "Hossam Eldin Shoukry";  
console.log(a.substring(a.length-8,a.length-13));
```



```
console.log(a.substr(-5,3)) // في هذه الدالة تضع فقط كم حرف تريد اظهار مع تحديد رقم  
اول حرف
```

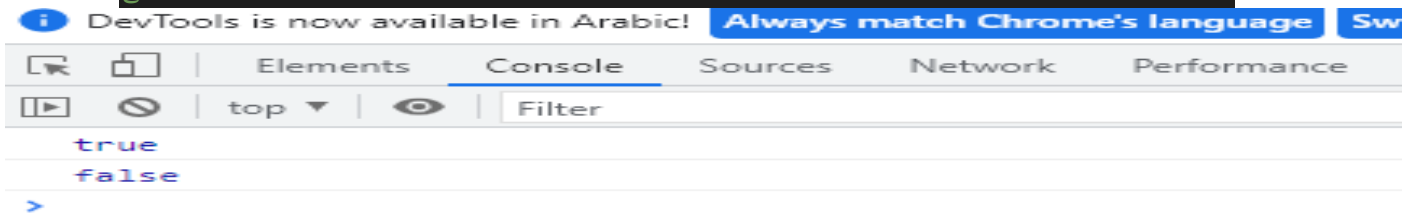

-:Includes الدالة

```
Let a= "Hossam Eldin Shoukry"
console.log(a.includes("Hossam")); // تختبر هذه الدالة ما تسأل عنه يعني هنا تختبر الدالة هل المتغير يحتوى على كلمة حسام ام لا
console.log(a.includes("Eldin", 8)) // تختبر الدالة ما اذا كانت كلمة الدين تبدأ من عند الحرف الثامن ام لا
```



-:StartWith الدالة

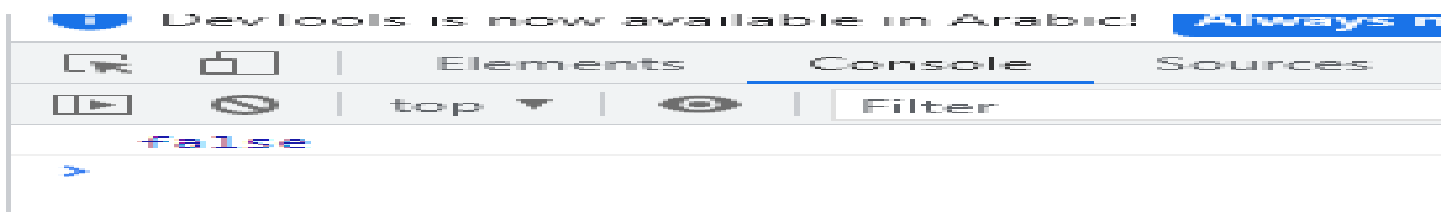
```
console.log(a.startsWith("H")); // هل قيمة المتغير تبدأ بحرف الاتش
console.log(a.startsWith("H",2)); // هل قيمة المتغير تبدأ بحرف الاتش بدأ من الحرف الثاني
```



-:endWith الدالة

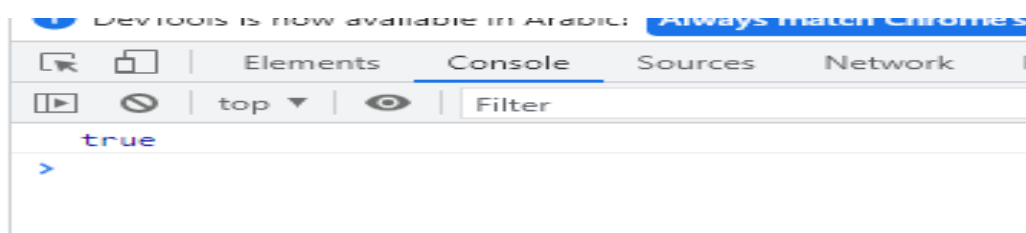
تختبر هل قيمة المتغير ت تنتهى بكذا وكذا ام لا بعد ما يتم حساب الطول -: lenght

```
console.log(a.endsWith("H")); // تنتهى بحرف الاتش ام لا
```



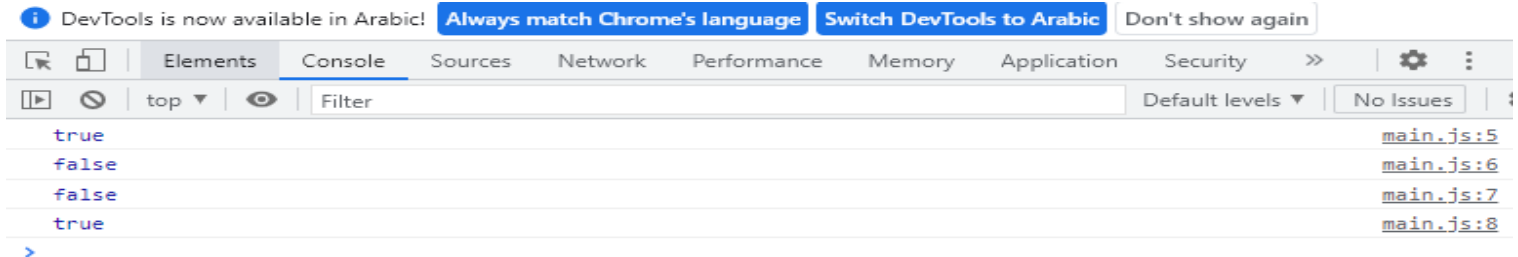
الان اذا اردنا تحديد طول المتغير بنفسنا مع الوضع فى الاعتبار انها -: not including end

```
console.log(a.endsWith("m",6)); //
```

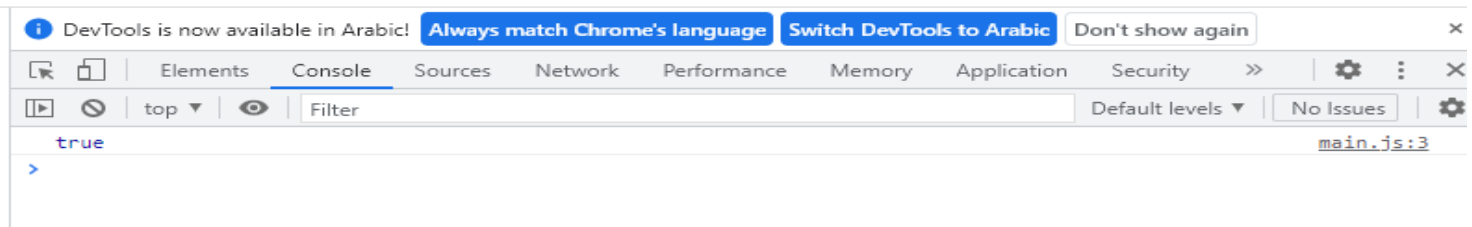


اليساوى واللايساوى قيمة ونصا (==, !=, ===, !==) :-

```
console.log("10"==10); // علامة اليساوى هنا تسأل عن القيمة فقط وليس نوع البيانات
console.log("10"!=10); // علامة اللايساوى هنا تسأل عن القيمة فقط وليس نوع البيانات
console.log("10"===10); // و نوع البيانات علامة اليساوى هنا تسأل عن القيمة
console.log("10"!==10); // و نوع البيانات علامة اللايساوى هنا تسأل عن القيمة
```

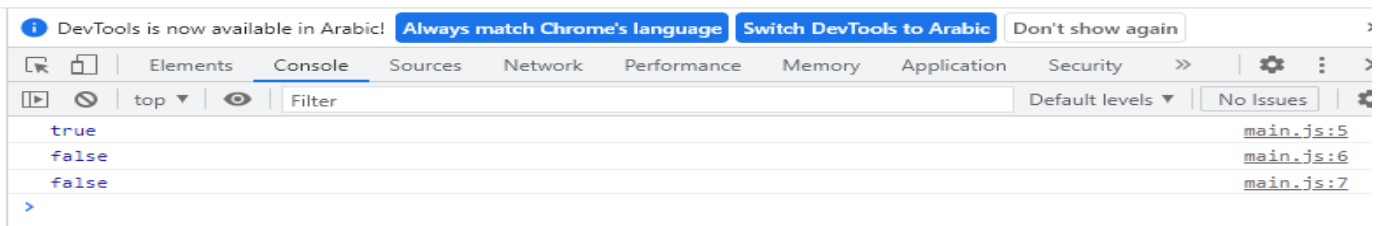


```
console.log(typeof("ikghikg")===typeof("pkpkp"));
```



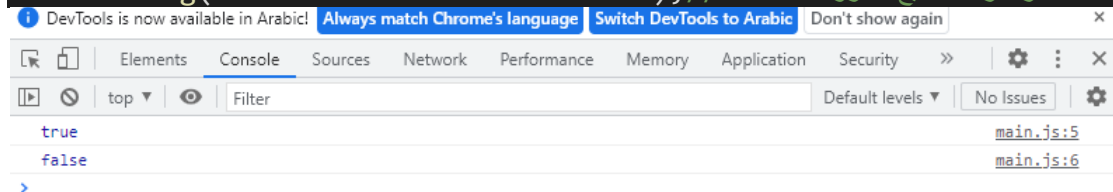
المثال السابق يسأل عن نوع البيانات فقط وليس القيمة.
الصحيح والخطأ:-

```
console.log(true); // اطبع علامة الصحيح
console.log(!true); // اطبع عكس علامة الصحيح وهى الخطأ
console.log(!(10=="10")); // الرقمين متساويين ولكن يوجد علامة ليس
```



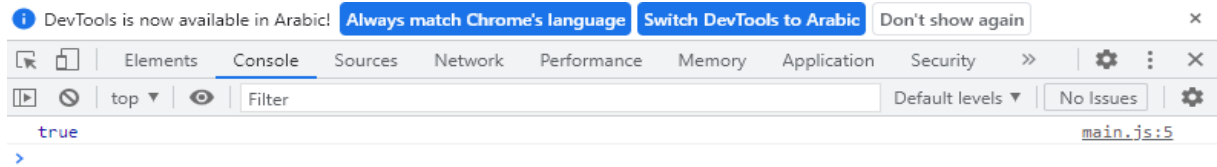
شرح الـ && And :-

```
console.log(10=="10" && 10>8 && 10<=20); // جميع الشروط متحققة
console.log(10=="10" && 10>8 && 10>=20); // ماعدا اخر شرط جميع الشروط متحققة
```



شرح الـ أو Or :-

```
console.log(10=="10" || 10<8 || 10>20); // تحقيق مطلب واح فقط من الثلاث مطالب
```



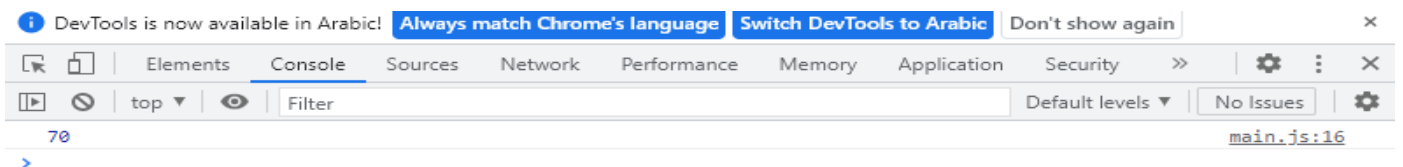
قاعدة لو If :-

```
let price = 100 ;
let discount=true;
let discountAmount=30;
let country="egypt";

if (discount === true){

    price -= discountAmount;
} else if(country==="egypt"){

    price-= 40;
}
console.log(price);
```



المثال التالي يوضح ماذا نعمل عند عدم تحقيق أي شرط باستخدام `else` فقط :-

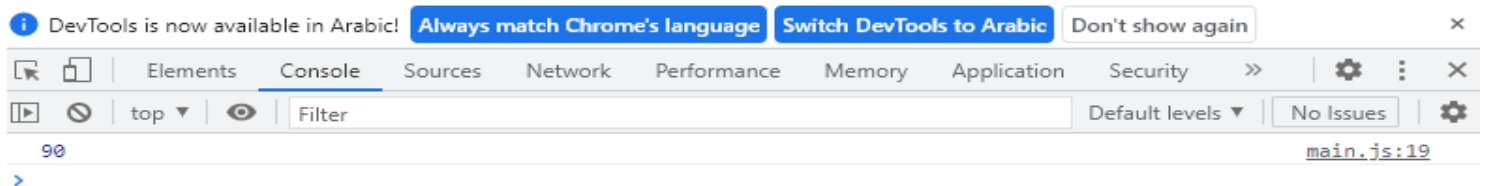
```
let price = 100 ;
let discount=true;
let discountAmount=30;
let country="egypt";

if (discount === false){

    price -= discountAmount;
} else if(country==="ksa"){

    price-= 40;

} else{
    price -= 10;
}
console.log(price);
```



شرح nested condition :-

هي باختصار عبارة توفر شرطين لتحقيق الطلب وهي تكون عبارة عن
`if` داخل `if` :-

```
let price = 100 ;
let discount=true;
let discountAmount=30;
let country="egypt";
let student=true;

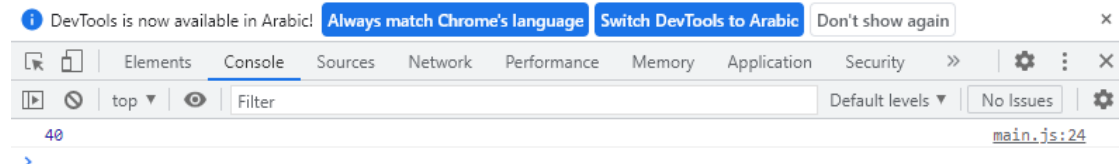
if (discount === false){

    price -= discountAmount;
} else if(country==="egypt"){

    if(student===true){
        price-= discountAmount+30;
    }
}
```

```
    }else{
        price-= discountAmount+10;
    }

} else{
    price -= 10;
}
console.log(price);
```



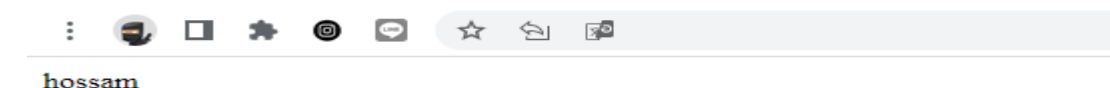
قاعدة الـ If المختصرة :-

```
// condition ? if true : if false
discount === true ? console.log("hossam"): console.log("ahmed");
```

مثال اخر على اعطاء قيمة متغير النتيجة التى سوف تخرجها الـ If :-

```
let result=discount === true ? "hossam":"ahmed";
document.write(result);
```

مع الوضع فى الاعتبار بازالة امر الكونسول من السطر البرمجى

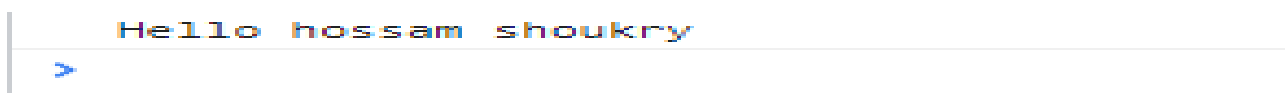


مثال اخر:-

```
console.log(discount === true ? "hossam" : "ahmed");
```

مثال اخر:-

```
let name="shoukry";
console.log(`Hello ${discount === true ? "hossam" : "ahmed"}
${name}`);
```



مثال على وضع اكثر من شرط في الـ If المختصرة :-

```
let price = 100 ;
let name="shoukry";
let discount=true;

discount=== false
? console.log("nothing")
: name === "ahmed"
? console.log("zero")
: price > 50
? console.log("that is right")
: console.log("unknown");
```

that is right

main.js:26

شرح Logical or :-

يتم بها ارجاع قيمة لمتغير كانت قيمته السابقة صفر او null :-

```
let price = 0;
console.log(`the price is ${price || 200}`); /*
1- اذا كانت قيمة المتغير صفر
2- undefined اذا كانت قيمة المتغير فارغة
3- false اذا كانت قيمة المتغير
4- null اذا كانت قيمة المتغير
5- اذا كانت قيمة المتغير نص فارغ-
*/ يتم ارجاع القيمة بالدالة || or
```

DevTools is now available in Arabic! Always match Chrome's language Switch DevTools to Arabic Don't show again

Elements Console Sources Network Performance Memory Application Security >>

top Filter Default levels No

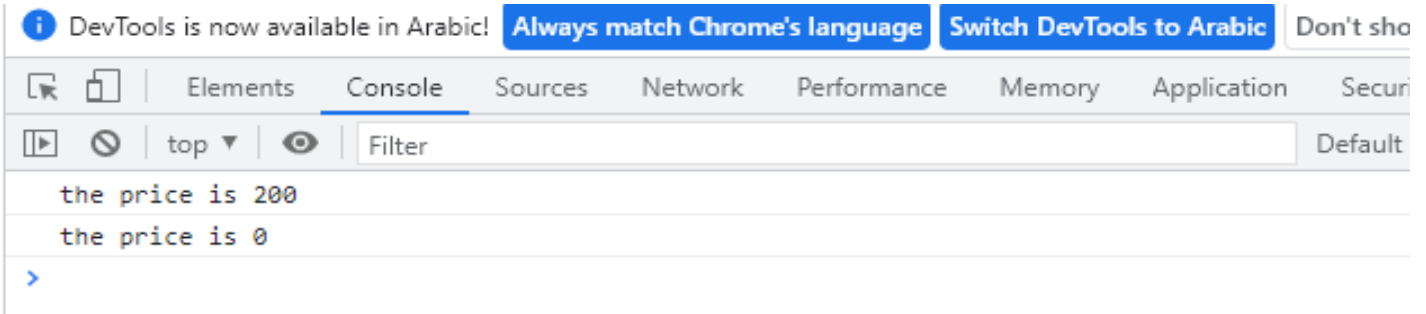
the price is 200

- قاعدة الـ Nulish coalescing operator??

لايقوم بارجاع الدالة التي تكون قيمتها false اى التي ترجع فى المخرجات false:-

لاحظ المثال التالى الفرق بين Logical or و Nulish coalescing operator??

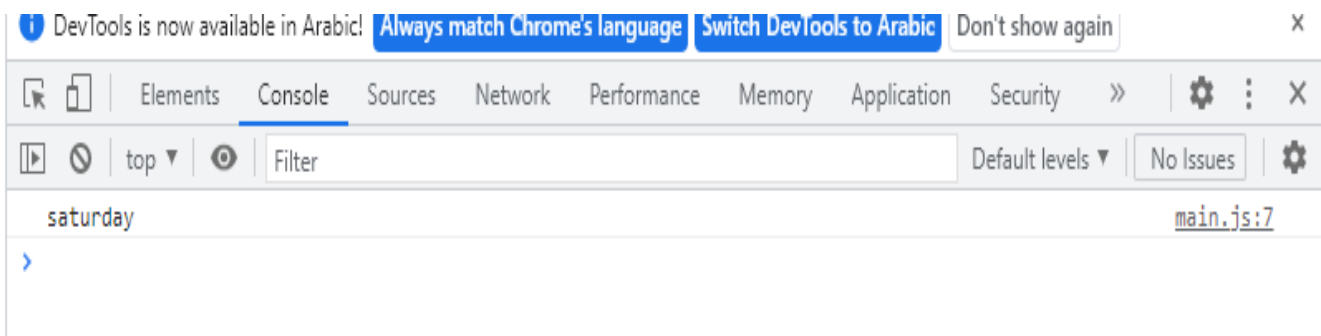
```
let price = 0;
console.log(`the price is ${price || 200}`);
console.log(`the price is ${price ?? 200}`);
```



-:Switch case شرح

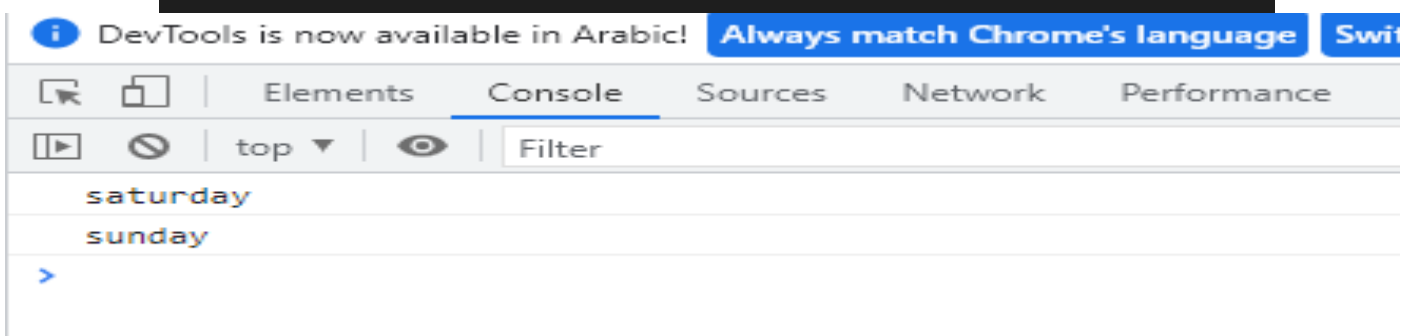
- 1- وظيفة البريك انها تقوم بايقاف عملية البحث عند العثور على الهدف المطلوب بمعنى انه اذا وجد قيمة اخرى نفس قيمة الكيز الاولى فلن يرجعه .
- مثال على ذلك:-

```
- let day = 0;
- switch (day){
- case 0:
- console.log("saturday");
- break;
- case 0:
- console.log("saturday");
- break;
- }
-
```



لاحظ المثال التالي عند حذف الـ **break** سيتم طباعة العنصرين في المخرجات :-

```
let day = 0;
switch (day){
case 0:
console.log("saturday");
case 0:
console.log("saturday");
break;
}
```



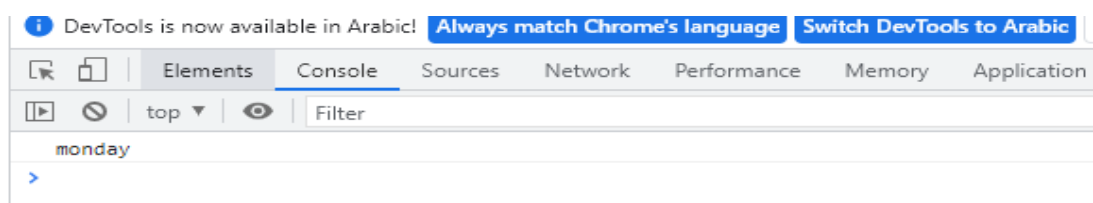
تتم عملية البحث في الـ **switch case** عن طريق الرقم الذي يكون مشترك بين المتغير والـ **case** وهو ما يسمى بالـ **Identical operator**.

المثال التالي يوضح عملية اظهار اليوم المطلوب:-

```
let day = 2;
switch (day){
case 0:
console.log("saturday");
break;

case 1:
console.log("sunday");
break;

case 2:
console.log("monday");
break;
}
```





إذا لم يكن هناك رقم مشترك بين الـ case والمتغير يتم استعمال مايسمى
بالـ default:-
المثال التالي المتغير يساوي 5 ولا يوجد اي case تساوي 5:-





```
let day = 5;
switch (day){
case 0:
console.log("saturday");
break;

case 1:
console.log("sunday");
break;

case 2:
console.log("monday");
break;
default:
console.log("fuck my life");
}
```

 DevTools is now available in Arabic! [Always match Chrome's language](#) [Switch DevTools to Arabic](#) [Don't show again](#)

  | Elements Console Sources Network Performance Memory Application

  | top ▼  | Filter 

fuck my life



المصفوفات Array

```
let myName = ["hossam", "ahmed", "mohamed"];
// يبدأ العد في المصفوفة من الرقم صفر يعنى لطباعة مثلا "اهلا احمد" تكون كالتالى
console.log(`Hello ${myName[0]}`);
console.log(`Hello ${myName[2]}`); // hello mohamed
console.log(`${myName[1][2]}`); // الوصول الى حرف m فى كلمة محمد
الـ
```

Elements	Console	Sources	Network	Performance	Memory	Application	Security	»	⚙	:	✕
▶	🔍	top ▼	👁	Filter	Default levels ▼	No Issues	⚙				
	Hello hossam										main.js:5
	Hello mohamed										main.js:6
	m										main.js:7

Nested Array

- هى عبارة عن مصفوفة داخل مصفوفة اخرى يمكن الوصول الى جميع عناصرها كالتالى :-

```
let myName = ["hossam", "ahmed", "mohamed" , ["ali" , "bassam"]];
console.log(`Hello ${myName[3]}`); // الوصول الى المصفوفة الصغيرة داخل المصفوفة
الكبيرة
console.log(`Hello ${myName[3][0]}`); // الوصول الى اسم على
console.log(`Hello ${myName[3][0][2]}`); // الوصول الى حرف الـ i فى اسم على
```

Hello ali,bassam	main.js:9
Hello ali	main.js:10
Hello i	main.js:11
>	

```
// تغيير عنصر من عناصر المصفوفة
// تغيير اسم احمد الى جمال
console.log(myName);
myName[1]="gamal";
console.log(myName);
لاحظ فى المخرجات الفرق بين المصفوفتين بعد التغيير
```

▶ (4) ['hossam', 'ahmed', 'mohamed', Array(2)]	main.js:16
▶ (4) ['hossam', 'gamal', 'mohamed', Array(2)]	main.js:18

/ تغيير مصفوفة داخل مصفوفة */*

```
let myName = ["hossam", "ahmed", "mohamed", ["ali", "bassam"]];  
myName[3] = ["sayed", "tal13at"];  
console.log(myName);
```

```
▼ (4) ['hossam', 'gamal', 'mohamed', Array(2)] ⓘ  
  0: "hossam"  
  1: "gamal"  
  2: "mohamed"  
▶ 3: (2) ['sayed', 'tal13at']
```

/ للاستفسار عن ما اذا كان المتغير لدينا هو مصفوفة ام لا */*

```
console.log(Array.isArray(myName));
```

true

main.js:24

>

تعامل المصفوفة مع الـ Length:-

مثلا اذا اردنا اضافة عنصر الى المصفوفة بطريقة برمجية دون الحاجة الى استعمال الاندكس
فسكون التعامل بطريقة الـ length:-

```
let myName = ["hossam", "ahmed", "mohamed" ];  
  
myName[3] = "hassan" // الطريقة القديمة لاضافة عنصر داخل المصفوفة  
  
myName[myName.length] = "gamal" // الطريقة الجديدة لاضافة عنصر داخل المصفوفة  
console.log(myName);
```

DevTools is now available in Arabic: [Always match Chrome's language](#) [Switch DevTools to Arabic](#) [Don't show again](#)

Elements Console Sources Network Performance Memory Application Security >> ⚙️ ⋮ ✕

▶ ⏏ top ▼ 🔍 Filter Default levels ▼ No Issues ⚙️

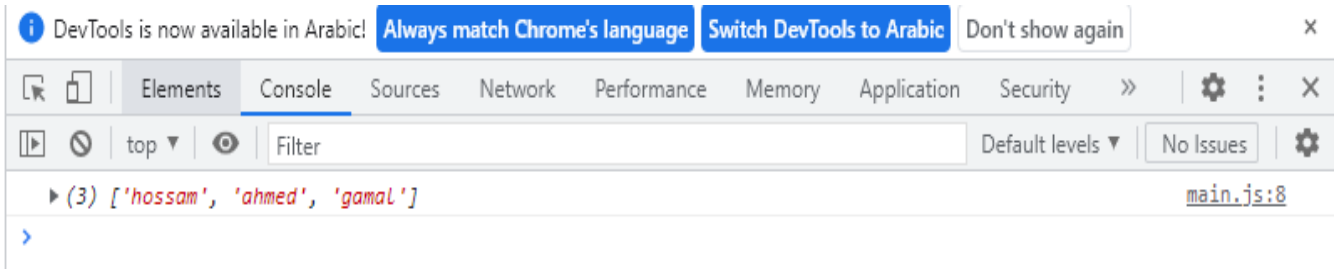
```
▶ (5) ['hossam', 'ahmed', 'mohamed', 'hassan', 'gamal']
```

main.js:9

>

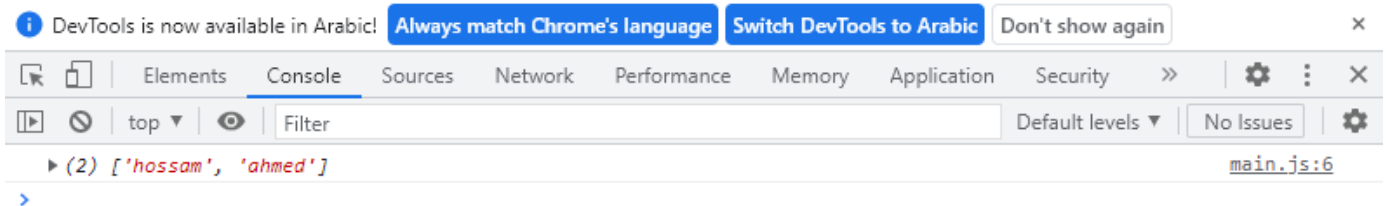
- لتحديث اخر عنصر في المصفوفة :-

```
let myName = ["hossam", "ahmed", "mohamed" ];  
myName[myName.length-1]="gamal" // الطريقة الجديدة لاضافة عنصر داخل  
المصفوفة  
console.log(myName);  
لاحظ ان كلمة محمد غيرت الى كلمة جمال
```



- لتحديد طول المصفوفة :-

```
let myName = ["hossam", "ahmed", "mohamed" ];  
myName.length=2; // حدد طول المصفوفة بكتابة اى رقم تريده  
console.log(myName);
```



طرق اضافة عناصر للمصفوفة

1. اضافة عناصر فى بداية المصفوفة نستعمل unshift:-

```
2. let myName = ["hossam", "ahmed", "mohamed" ];  
3. console.log(myName);  
4. myName.unshift("ali" , "nabil");  
5. console.log(myName);
```



2. اضافة عناصر فى نهاية المصفوفة باستعمال push :-

```
myName.push("samah" , "enas");  
console.log(myName);
```

```
▶ (7) ['ali', 'nabil', 'hossam', 'ahmed', 'mohamed', 'samah', 'enas']
```

>

3. حذف اول عنصر من المصفوفة:-

```
let myName = ["hossam","ahmed","mohamed" ];  
myName.shift();  
console.log(myName);
```

```
▶ (7) ['ali', 'nabil', 'hossam', 'ahmed', 'mohamed', 'samah', 'enas']
```

```
▶ (6) ['nabil', 'hossam', 'ahmed', 'mohamed', 'samah', 'enas']
```

>

4. حذف وتخزين اول عنصر فى المصفوفة داخل المتغير :-

```
let first = myName.shift(); // حذف العنصر ووضعه داخل متغير  
console.log(myName); // طباعة المصفوفة وحذف العنصر  
  
console.log(`the first name is ${first}`); // عرض العنصر المحذوف داخل  
متغير
```

```
▶ (7) ['ali', 'nabil', 'hossam', 'ahmed', 'mohamed', 'samah', 'enas']
```

```
▶ (6) ['nabil', 'hossam', 'ahmed', 'mohamed', 'samah', 'enas']
```

```
the first name is ali
```

5. حذف وتخزين اخر عنصر فى المصفوفة :-

```
6. let last = myName.pop(); // حذف العنصر الاول ووضعه داخل متغير
7. console.log(myName); // طباعة المصفوفة وحذف العنصر
8.
9. console.log(`the last name is ${last}`); // عرض العنصر المحذوف
    داخل متغير
```

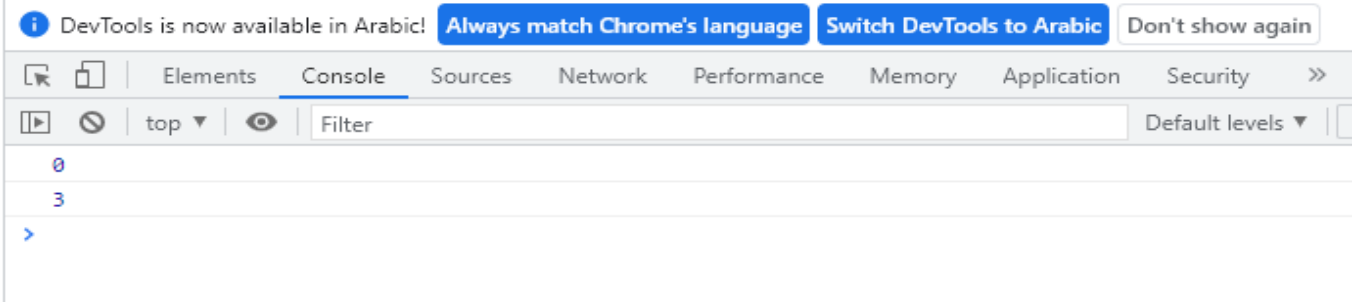
```
► (7) ['ali', 'nabil', 'hossam', 'ahmed', 'mohamed', 'samah', 'enas']
► (6) ['ali', 'nabil', 'hossam', 'ahmed', 'mohamed', 'samah']
the last name is enas
```

>

البحث فى المصفوفة باستخدام Index Of و Last Index Of :-

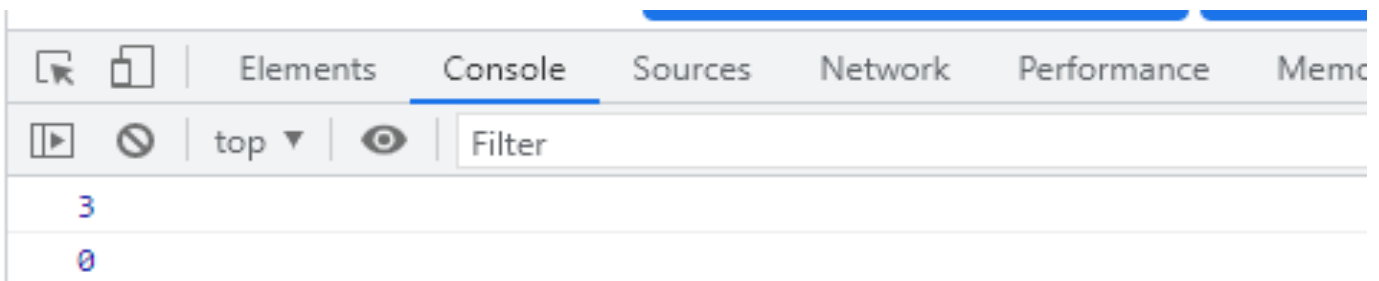
1. Index of :- تبحث فى المصفوفة بدأ من اليسار الى اليمين كالتالى :-

```
let myName = ["hossam", "ahmed", "mohamed", "hossam"];
console.log(myName.indexOf("hossam"));
console.log(myName.indexOf("hossam", 2)); // ابحث عن كلمة حسام بدأ من الاسم
    الثانى فى المصفوفة
console.log(myName.indexOf("hossam") === -1); // هل اسم حسام غير موجود ؟
    والمخرجات ستكون false لان اسم حسام موجود
```



2. Last Index of :- تبحث فى المصفوفة من اليمين الى اليسار كالتالى :-

```
console.log(myName.lastIndexOf("hossam"));
console.log(myName.lastIndexOf("hossam", -2)); // ابحث عن كلمة حسام بدأ من
```



الاسم الثاني في المصفوفة ولكن ابدأ العد من يمين المصفوفة

3. دالة الـ include تسأل هل العنصر موجود ام لا :-

```
let myName = ["hossam", "ahmed", "mohamed", "hossam"];
console.log(myName.includes("hossam"));
console.log(myName.includes("hossam", 3)); // هل اسم حسام موجود بدأ العد من الاسم الرابع
```

true main.js:1

true main.js:1

تجربة index of , last index of داخل الشرط if

```
if (myName.indexOf("osama") === -1) { // اذا كان اسم اسامة غير موجود اطبع انه غير موجود
    console.log("not found");
}
```

not found main.js:16

طريقة ترتيب المصفوفة باستعمال الدالة .sort()

```
let myName = [10, "hossam", "mohammed", "90", 1000, 100, 20, "10", -20, -10];
console.log(myName);
console.log(myName.sort());
```

top Filter

▶ (10) [10, 'hossam', 'mohammed', '90', 1000, 100, 20, '10', -20, -10]

▶ (10) [-10, -20, 10, '10', 100, 1000, 20, '90', 'hossam', 'mohammed']

رقم 1000 أتى في الترتيب قبل رقم 20 لان الدالة ترتب من الرقم الاول وليس من الثاني واول رقم في العدد 1000 يحتوى على واحد 1 ام العدد 20 اول رقم فيه يحتوى على 2 . ثم بعد عد اول رقم من العدد يبدأ بالعد من تاني رقم وهذا يعنى انه اذا تم استبدال العدد 1000 بـ 2000 سيتم كتابة العدد 2000 بعد العدد 20 وليس قبله وكذلك الحال مع العدد 90 اذا تم استبداله بـ 9000.

▶ (10) [10, 'hossam', 'mohammed', '9000', 2000, 100, 20, '10', -20, -10]

▶ (10) [-10, -20, 10, '10', 100, 20, 2000, '9000', 'hossam', 'mohammed']

دالة reverse :- تقوم دالة الـ reverse بعكس اخر دالة تم طباعتها يعنى ذلك انه سيتم عكس دالة الـ myName.sort واذا كانت دالة الـ myName هي اخر دالة تم طباعتها فان دالة الـ reverse ستقوم بعكسها.

```
let myName = [10 , "hossam" , "mohammed" , "9000", 2000 , 100 , 20, "10", -20 , -10 ];
console.log(myName);
console.log(myName.sort());
console.log(myName.reverse());
```

```
▶ (10) [10, 'hossam', 'mohammed', '9000', 2000, 100, 20, '10', -20, -10]
▶ (10) [-10, -20, 10, '10', 100, 20, 2000, '9000', 'hossam', 'mohammed']
▶ (10) ['mohammed', 'hossam', '9000', 2000, 20, 100, '10', 10, -20, -10]
```

Slice Method

تعامل دالة الـ slice مع المصفوفة Array

```
let myName = [10 , "hossam" , "mohammed" , "9000", 2000 , 100 , 20, "10", -20 , -10 ];
console.log(myName.slice()); // طباعة المصفوفة كاملة
console.log(myName.slice(2)); // طباعة المصفوفة بدأ من العنصر الثالث
console.log(myName.slice(1,4)); // not including end طباعة من اول عنصر الى ثالث عنصر علما بأنها لا تشمل النهاية
```

DevTools is now available in Arabic! Always match Chrome's language Switch DevTools to Arabic Don't show again

Elements Console Sources Network Performance Memory Application Security >> | ⚙️ | ⋮

top Filter Default levels No Issues

```
▶ (10) [10, 'hossam', 'mohammed', '9000', 2000, 100, 20, '10', -20, -10] main.js:5
▶ (8) ['mohammed', '9000', 2000, 100, 20, '10', -20, -10] main.js:6
▶ (3) ['hossam', 'mohammed', '9000'] main.js:7
```

```
let myName = [10 , "hossam" , "mohammed" , "9000", 2000 , 100 , 20, "10", -20 , -10 ];
console.log(myName.slice(-7)); // تحديد رقم البداية بالعكس ولكن العد يكون من اليسار الى اليمين
console.log(myName.slice(1,-7)); // not including end طباعة المصفوفة بدأ من العنصر الثانى ونهاية بالعنصر السابع بالعكس ولا يشمل النهاية كما هو معروف
```



```
console.log(myName.slice(-5,-1)); //not including end
```

‣ (7) ['9000', 2000, 100, 20, '10', -20, -10]

‣ (2) ['hossam', 'mohammed']

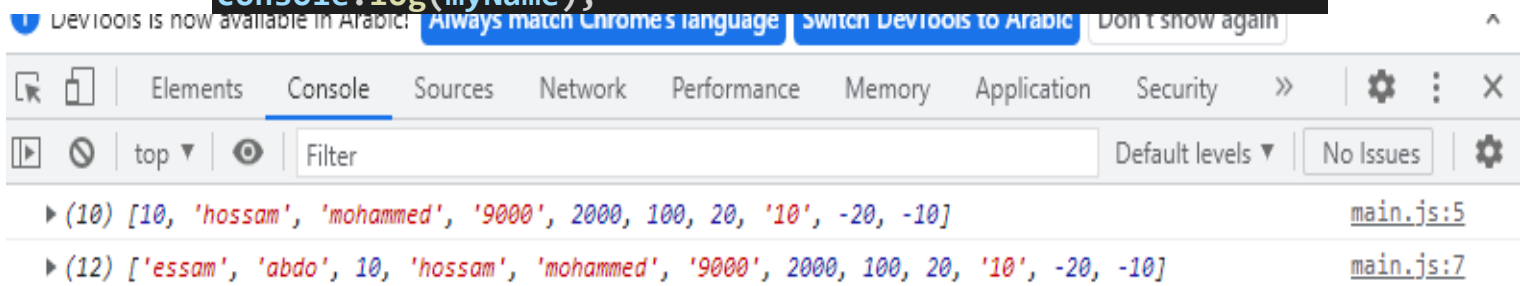
‣ (4) [100, 20, '10', -20]



دالة splice

تقوم بالحذف والاضافة فى المصفوفة (0,0,sammah,osama) الرقم الاول فى القوس هو رقم العنصر الذى ستضع عنده التعديلات فى المصفوفة والرقم الثانى فى القوس هو عدد العناصر التى ستقوم بحذفها والثالث والرابع هى العناصر التى ستقوم باضافتها ويتضح ذلك بالمثال التالى:-

```
let myName = [10 , "hossam" , "mohammed" , "9000", 2000 , 100 , 20, "10", -20 , -10 ];
console.log(myName);
myName.splice(0,0,"essam","abdo");
console.log(myName);
```



لاحظ المثال التالى انه قد تم العدد 10 من المصفوفة لانه قد تم تحديد حذف عنصر من الاندكس الاول

```
let myName = [10 , "hossam" , "mohammed" , "9000", 2000 , 100 , 20, "10", -20 , -10 ];
console.log(myName);
myName.splice(0,1,"essam","abdo");// حذف عنصر من الاندكس الاول واطافة الاسمين
التاليين
console.log(myName);
```

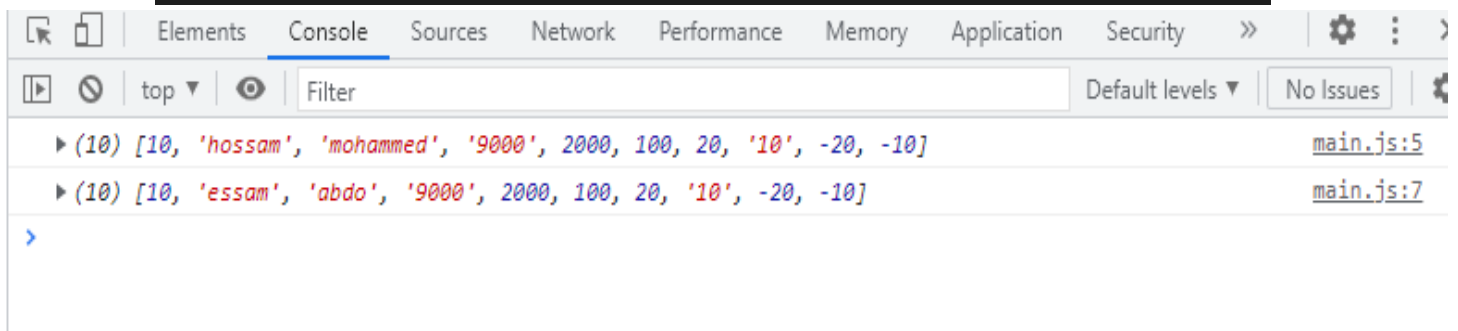
‣ (10) [10, 'hossam', 'mohammed', '9000', 2000, 100, 20, '10', -20, -10]

‣ (11) ['essam', 'abdo', 'hossam', 'mohammed', '9000', 2000, 100, 20, '10', -20, -10]



المثال التالي يوضح حذف عنصريين وإضافة الاسماء الموجودة في المثال بدأ من الاندكس الثاني.

```
let myName = [10 , "hossam" , "mohammed" , "9000", 2000 , 100 , 20, "10", -20 , -10 ];
console.log(myName);
myName.splice(1,2,"essam","abdo");// حذف عنصر من الاندكس الاول وإضافة الاسمين التاليين
console.log(myName);
```

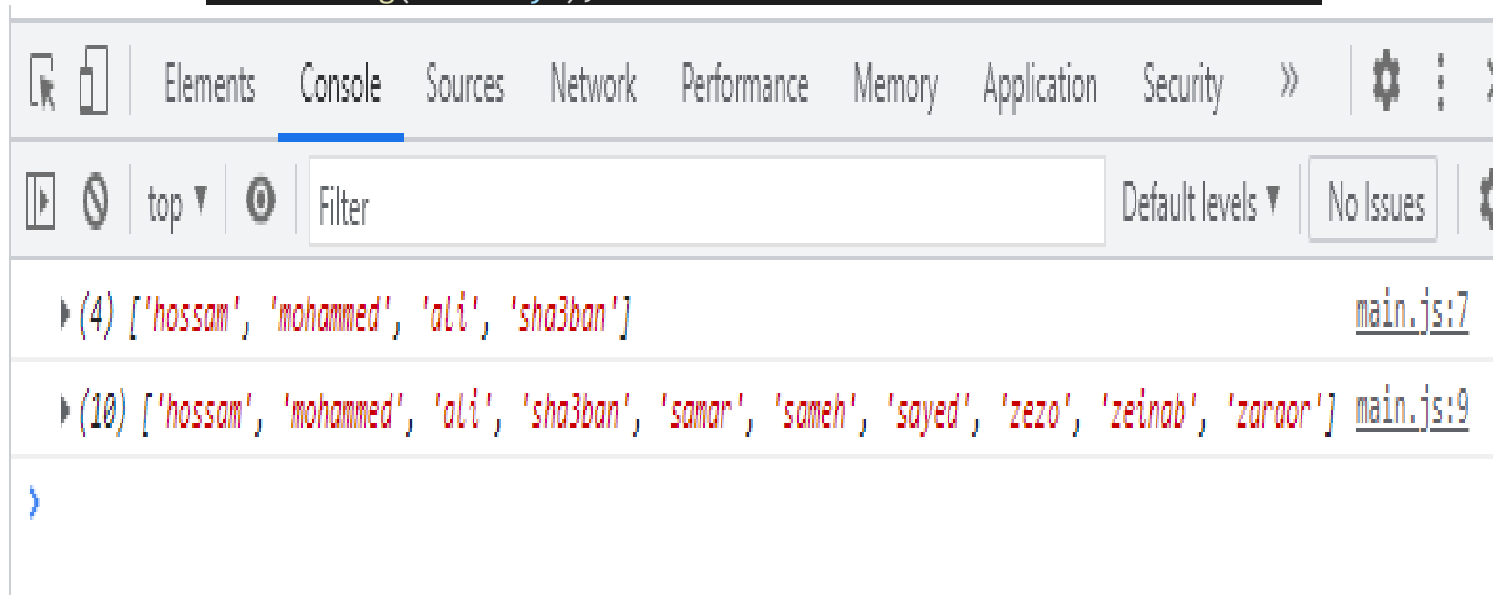


دالة concat

تقوم هذه الدالة بالربط بين البيانات المثال التالي يوضح ربط مصفوفتين مع بعضهما ووضعهما في مصفوفة واحدة .

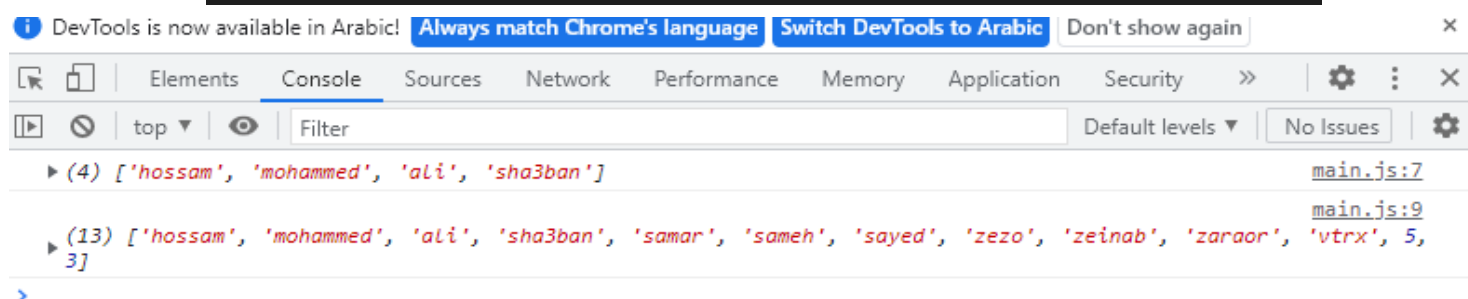
```
let myName = [ "hossam" , "mohammed" , "ali" , "sha3ban"];
let myFriends=["samar","sameh", "sayed"];
let myFamily=["zezo","zeinab","zaraor"]
console.log(myName);
let allArrays = myName.concat(myFriends,myFamily);// جمع المصفوفات
// ووضعهما في مصفوفة واحدة
```

```
console.log(allArrays);
```



كما يمكن اضافة عناصر زيادة على المصفوفات الموجودة من خلال كتابة ماتريد زيادته فى الاقواس كما فى المثال التالى :-

```
let allArrays = myName.concat(myFriends,myFamily, "vtrx" ,  
[5,3]);  
console.log(allArrays);
```



دالة الـ Join تقوم بدمج عناصر المصفوفة مع بعض والفصل بينهم بفصل انت الذى تقوم بتحديدده كما فى المثال التالى:-

```
console.log(allArrays.join(" ")); // الفصل بين عناصر المصفوفة بفراغ  
console.log(allArrays.join("@")); // الفصل بين عناصر المصفوفة بعلامة  
معينة  
console.log(allArrays.join("|")); // ٠٠.....  
.....
```

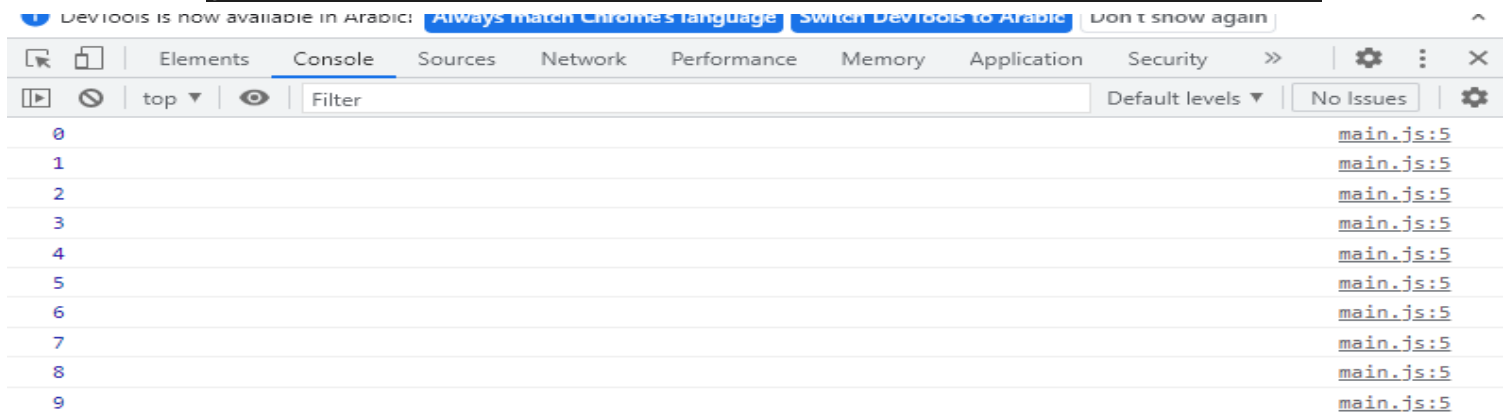
```
console.log(allArrays.join(" ").toLocaleUpperCase());//  
تحويل حروف المصفوفة الى حروف كابتل
```

hossam mohammed ali sha3ban samar sameh sayed zezo zeinab zaraor vtrx 5 3	main.js:10
hossam@mohammed@ali@sha3ban@samar@sameh@sayed@zezo@zeinab@zaraor@vtrx@5@3	main.js:11
hossam mohammed ali sha3ban samar sameh sayed zezo zeinab zaraor vtrx 5 3	main.js:12
HOSSAM MOHAMMED ALI SHA3BAN SAMAR SAMEH SAYED ZEZO ZEINAB ZARAOR VTRX 5 3	main.js:13

دالة for ، Loop

تقوم هذه الدالة بتكرار عناصر المصفوفة حسب الشروط الموجودة.

```
for(i=0 ; i<10 ; i++) { // تحديد الرقم الذى سيبدأ العد من عنده ونهاية العد وعدد الزيادة فى كل  
مرة  
    console.log(i)  
}
```

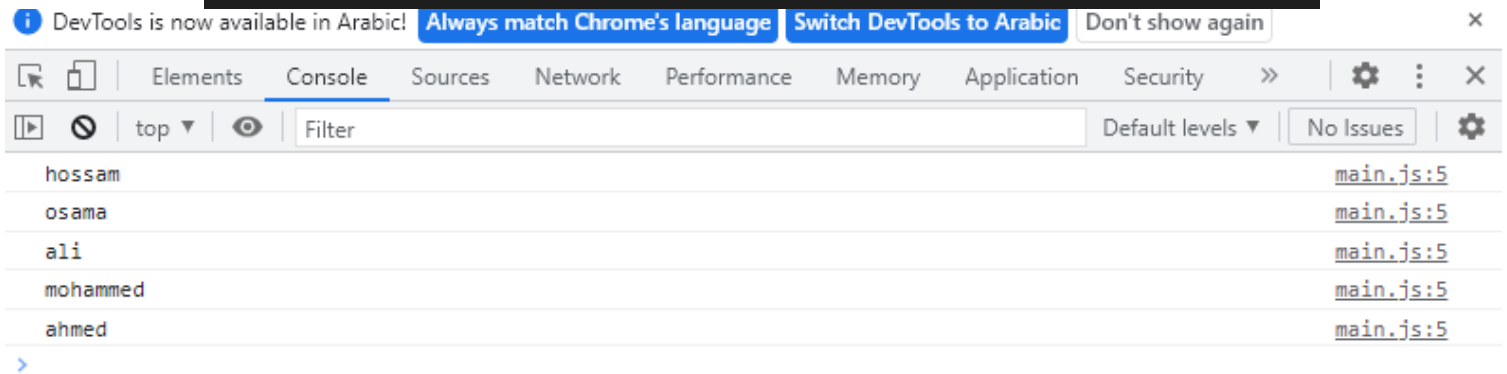


تطبيق الدالة على المصفوفة لفرز عناصر المصفوفة :- مع الوضع فى الاعتبار انه يجب
تحديد عدد عناصر المصفوفة او طولها داخل دالة الـ for

```
let array=["hossam" , "osama" , "ali" , "mohammed", "ahmed"]  
for(i=0 ; i<5 ; i++) { // تطبيق الدالة على المصفوفة  
    console.log(array[i])  
}
```

كتابة الكود بشكل افضل حتى لا يتم حدوث خطأ حيث سيتم استبدال عدد عناصر المصفوفة بتحديد طولها ديناميكيا
كما في المثال التالي

```
let array=["hossam" , "osama" , "ali" , "mohammed", "ahmed"]
for(i=0 ; i<array.length ; i++) { // تطبيق الدالة على المصفوفة
    console.log(array[i])
}
```



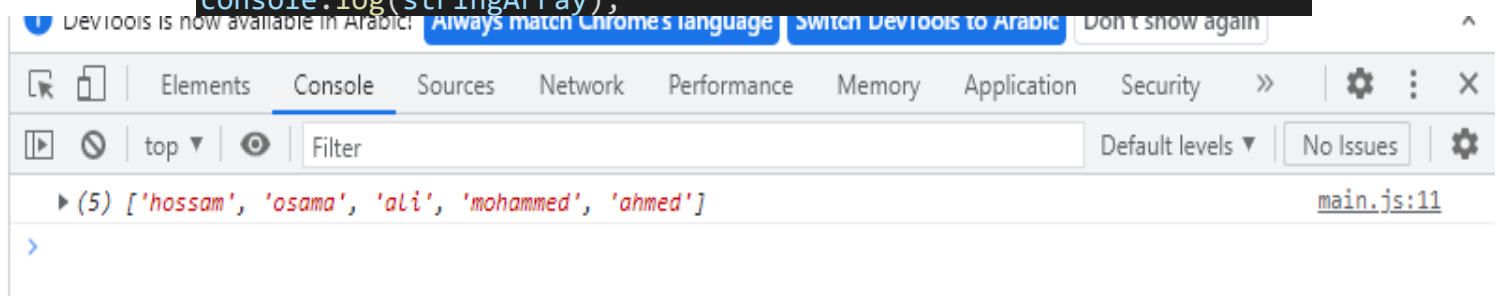
```
/* انشاء مصفوفة جديدة يضاف فيها عناصر المصفوفة الاولى التي تكون من نوع نصي */
let array =[10,5,6,"hossam" , "osama" , "ali" , "mohammed",
"ahmed"]
let stringArray= []

for(i=0 ; i<array.length ; i++) {

    if(typeof array[i]=== 'string'){
        stringArray.push(array[i]);
    }

}

console.log(stringArray);
```



Nested For Loop

فور داخل فور المثال التالي يشرح الية استخدام دالة الـ for لعملية انشاء منتج وخصائصه.

```
let products =["keyboard", "Mouse" , "Pen" , "Pad"  
 , "Monitor"];  
let colors=["Red", "Green" , "Black"]  
let models =[2020,2021];  
for(let i =0;i<products.length;i++){  
  console.log("#".repeat(15))  
  console.log(`the product is ${products[i]}`)  
  console.log("#".repeat(15))  
  for(j=0; j<colors.length;j++){  
    console.log(colors[j])  
  
  }  
  for(m=0; m<models.length;m++){  
    console.log(models[m])  
  
  }  
}
```

DevTools is now available in Arabic! Always match Chrome's language Switch DevTools to Arabic Don't show again		x	
		Elements	Console
		Sources	Network
		Performance	Memory
		Application	Security
		>>	
		top ▾	Filter
		Default levels ▾	No Issues
		⚙️	
#####		main.js:6	⬆
the product is keyboard		main.js:7	
#####		main.js:8	
Red		main.js:10	
Green		main.js:10	
Black		main.js:10	
2020		main.js:15	
2021		main.js:15	
#####		main.js:6	
the product is Mouse		main.js:7	
#####		main.js:8	
Red		main.js:10	
Green		main.js:10	
Black		main.js:10	
2020		main.js:15	
2021		main.js:15	
#####		main.js:6	
the product is Pen		main.js:7	
#####		main.js:8	
Red		main.js:10	
Green		main.js:10	
Black		main.js:10	
2020		main.js:15	
2021		main.js:15	
#####		main.js:6	
the product is Pad		main.js:7	
#####		main.js:8	⬇
Red		main.js:10	
Green		main.js:10	
Black		main.js:10	
2020		main.js:15	
2021		main.js:15	
#####		main.js:6	
the product is Monitor		main.js:7	
#####		main.js:8	
Red		main.js:10	
Green		main.js:10	
Black		main.js:10	
2020		main.js:15	
2021		main.js:15	

تعامل الـ for مع الـ Break و الـ Lable و الـ continue

1. تعامل الـ Break:- تقوم بإيقاف عمل الـ For عند وضعها في شرط كالمثال التالي الذي يشرح إيقاف طباعة باقى العناصر اذا كان الدور على طباعة العنصر الثالث.

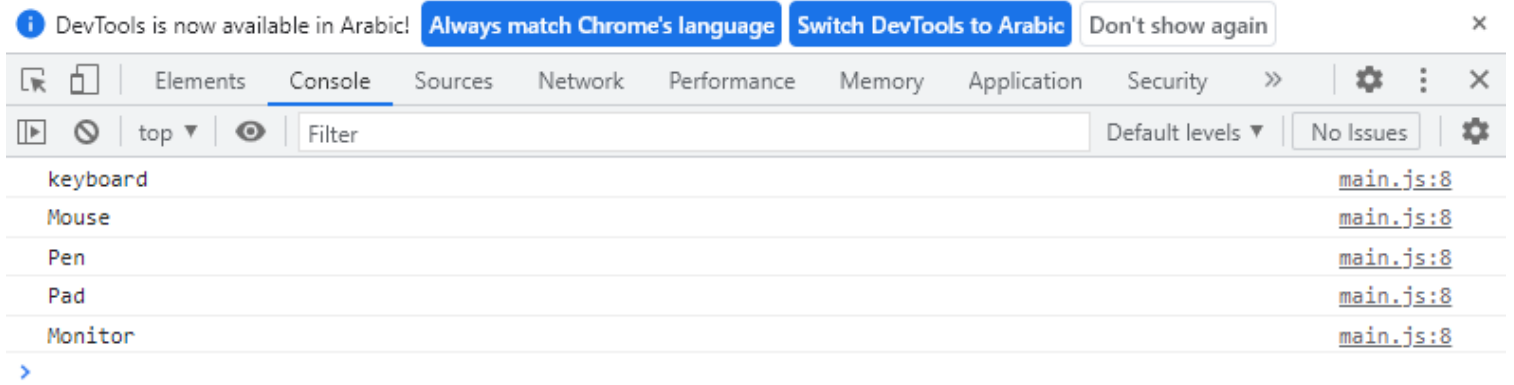
```
2.let products=["keyboard", "Mouse" , "Pen" ,  
    "Pad" , "Monitor"];  
3.let colors=["Red", "Green" , "Black"]  
4.for(i=0;i<products.length;i++){  
5.  
6.    console.log(products[i])  
7.    if (products[i]==="Pen"){  
8.        break;  
9.    }  
10.  
11. }
```

Filter	Default levels ▼	No Issues	⚙
keyboard		main.js:5	
Mouse		main.js:5	
Pen		main.js:5	

لاحظ ان ترتيب الكود مهم جدا حيث انك تقوم بتنفيذ طباعة العنصر الذى يأتى دوره فى اللوب اولاً ثم بعد ذلك تفحص ما اذا كان هذا العنصر يساوى الشرط ام لا لذلك تم كتابة الكود `console.log(products[i])` اولاً .

المثال الثانى يوضح تعامل الـ for مع الـ continue وهى التى تقوم بعمل حذف للعناصر المحددة فى الشرط وعدم طباعتها .

```
et products=["keyboard", 10,"Mouse" ,20, "Pen" ,30, "Pad" ,  
    "Monitor"];  
let colors=["Red", "Green" , "Black"]  
for(i=0;i<products.length;i++){  
  
    if (typeof products[i]==="number"){  
        continue;  
    }  
    console.log(products[i])  
}
```

تم كتابة كود `console.log(product[i])` في الاخر لانه اذا كتبت قبل الشرط سيتم طباعة الارقام قبل ان يتم فحص نوعها اذا كانت رقم ام لا وبالتالي لفائدة.

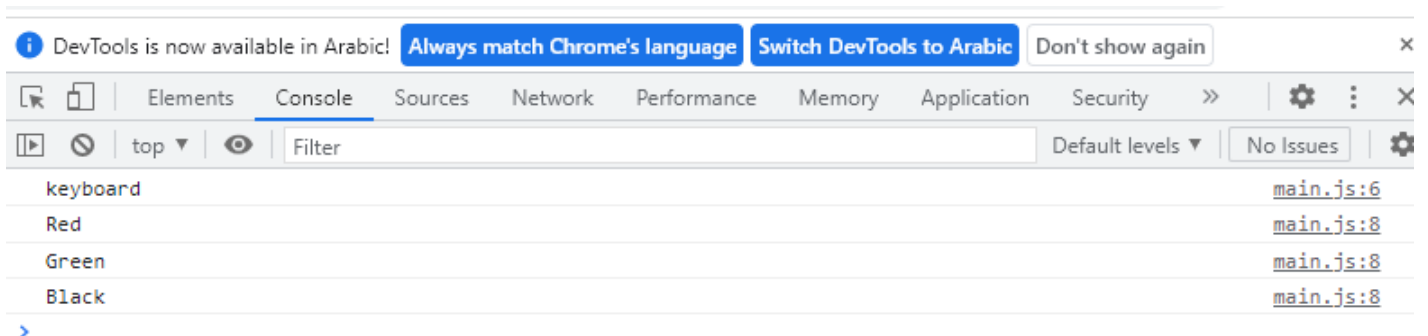
3. الـ `Lable` تقوم بعمل تعريف للـ `for` الرئيسية و كذلك الفرعية بحيث يمكن تطبيق الشرط على العناصر من خلال هذا التعريف دون الحاجة الى كتابة اسم الـ `for` نفسها.

- المثال التالي يقول اذا كان العنصر هو اللون الاسود قم بعمل ايقاف للوب الرئيسية:-

```
let products=["keyboard","Mouse" , "Pen" , "Pad" , "Monitor"];
let colors=["Red", "Green" , "Black"]

mainFoor:for(i=0;i<products.length;i++){
  console.log(products[i])
  nestedFoor: for(j=0;j<colors.length;j++){

    console.log(colors[j])
    if(colors[j]==="Black"){
      break mainFoor;
    }
  }
}
```



لاحظ انه لم يقوم بطباعة باقى عناصر الـ `mainFoor`

مثال على طباعة عناصر وألوانها باستخدام الـfor والhtml

```
let products =["keyboard","Mouse" , "Pen" , "Pad" , "Monitor"];
let colors=["Red", "Green" , "Black"]
showCount=5;
document.write(`<h1>show ${showCount} Products</h1>`)

for(i=0;i<showCount;i++){
document.write(`<div>`)
document.write(`<h2> [${i+1}] .. ${products[i]} </h2>`)
for(j=0;j<colors.length;j++){
    document.write(`<p> ${colors[j]} </p>`)
}
document.write(`<p> ${colors.join("|")} </p>`)
    document.write(`<div>`)
}
```

show 5 Products

[1] .. keyboard

Red
Green
Black
Red|Green|Black

[2] .. Mouse

Red
Green
Black
Red|Green|Black

[3] .. Pen

Red
Green
Black
Red|Green|Black

[4] .. Pad

Red
Green
Black
Red|Green|Black

[5] .. Monitor

Red
Green
Black
Red|Green|Black

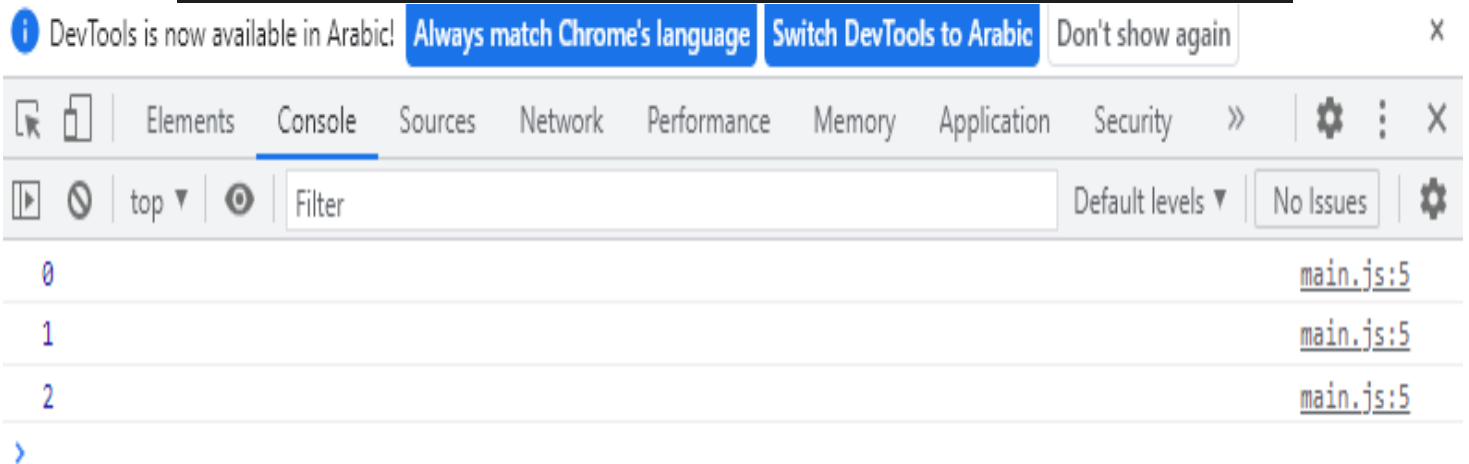
While Loop

While Loop: تقوم بنفس عمل الـ for ولكن تختلف في كتابة النص البرمجي كما في المثال التالي:-

```
let products = ["keyboard", "Mouse" , "Pen" , "Pad" , "Monitor"];
let index = 0;

while(index<10){
  console.log(index);
  index++;

  if(index===3){
    break;
  }
}
```



لاحظ هنا انه لم يتم طباعة الرقم 3 على الرغم من كتابة النص `index++` قبل `console.log(index)` في الـ `If condition` لانه تم كتابة `index++` بعد النص `console.log(index)` على عكس `for loop` حيث يتم كتابة `index++` قبل نص الطباعة `console.log(index)` وبالتالي سيتم فحص الرقم قبل طباعته.

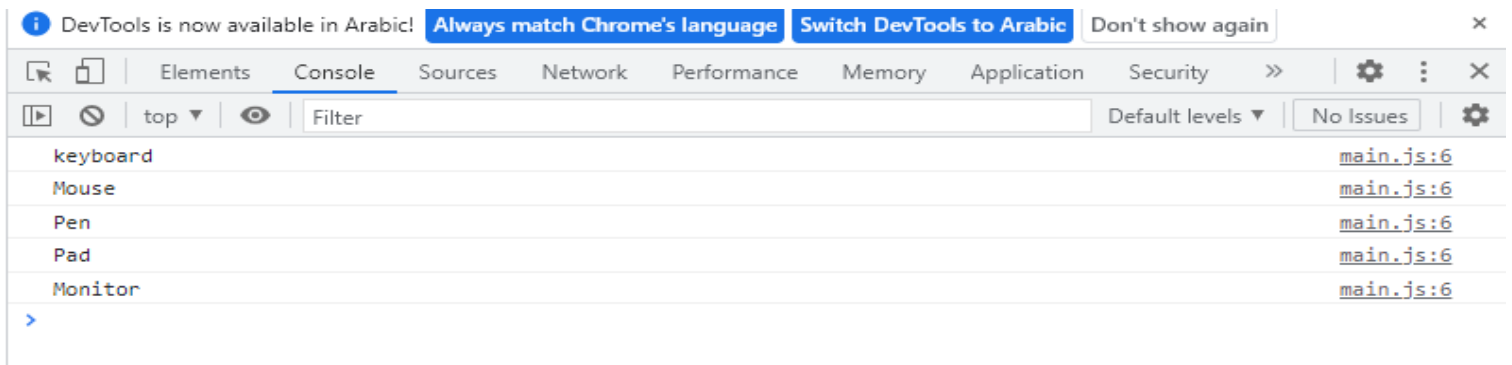
مثال على طباعة عناصر المصفوفة بالـ while:-

```
let products =["keyboard","Mouse" , "Pen" , "Pad" , "Monitor"];
let index = 0;

while(index<products.length){

    console.log(products[index]);
    index++;

}
```



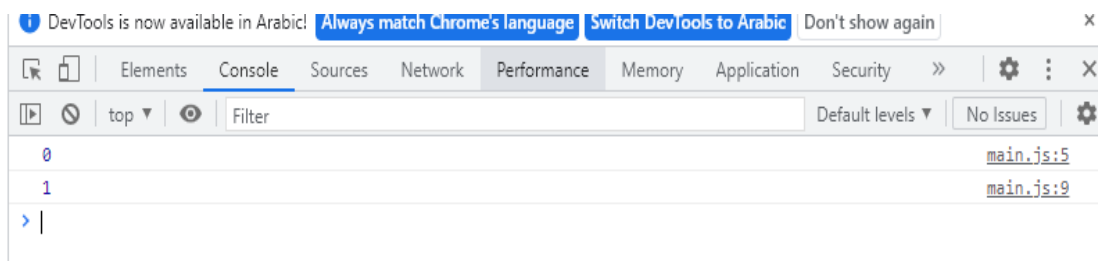
من الطرق الاخرى لعمل Loop هي طريقة do while

تقوم بعمل تنفيذ للسطر البرمجي اولاً قبل تنفيذ الشرط عليه حيث يتم تنفيذ do اولاً ثم while كما في المثال التالي الذي يقوم بتنفيذ الامر قبل اجراء عملية التحقق:-

```
let products =["keyboard","Mouse" , "Pen" , "Pad" , "Monitor"];
let index = 0;
do{

    console.log(index)
    index++

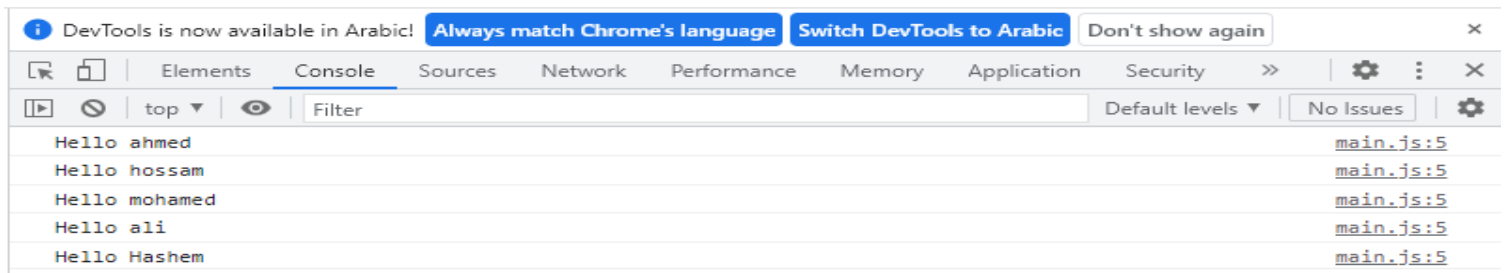
}while (false){
    console.log(index)
}
```



دالة الـ Function

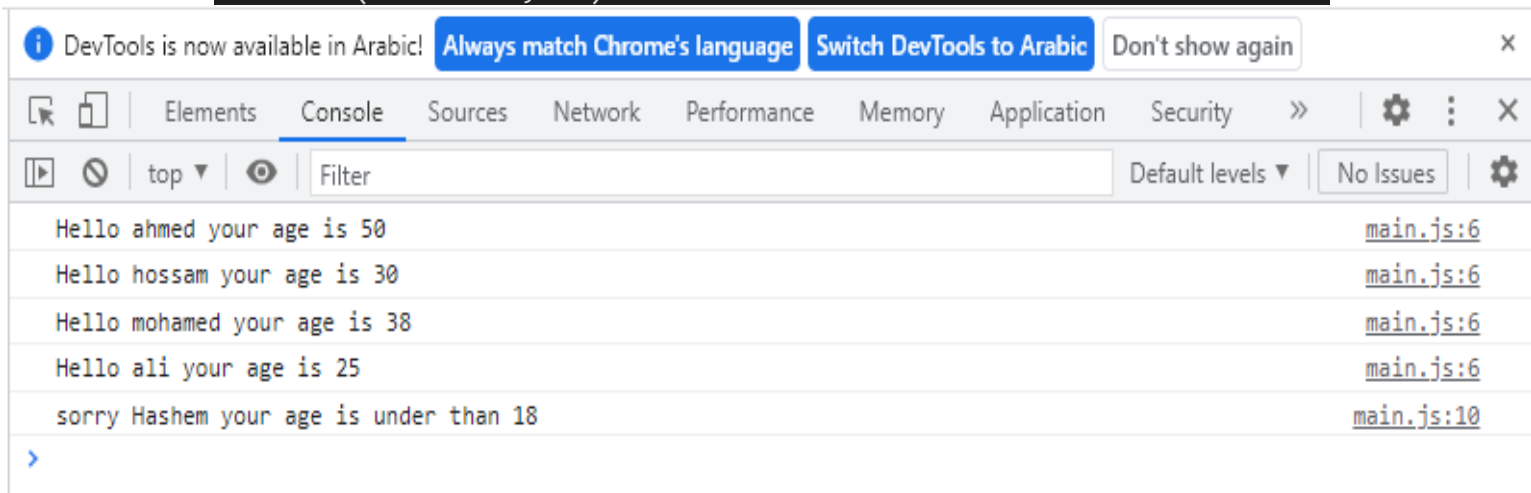
```
function test(user){ // هنا يكتب المتغير بين الاقواس ()
    console.log(`Hello ${user}`)

} test("ahmed") // هنا يكتب قيمة المتغير بين الاقواس ()
test("hossam")
test("mohamed") /* يمكن عمل اكثر من قيمة لمتغير واحد */
test("ali")
test("Hashem")
```



```
function test(user , age){ // يمكن وضع اكثر من متغير
if(age > 20){
    console.log(`Hello ${user} your age is ${age}`)
}
else{
    console.log(`sorry ${user} your age is under than ${age}`)
}

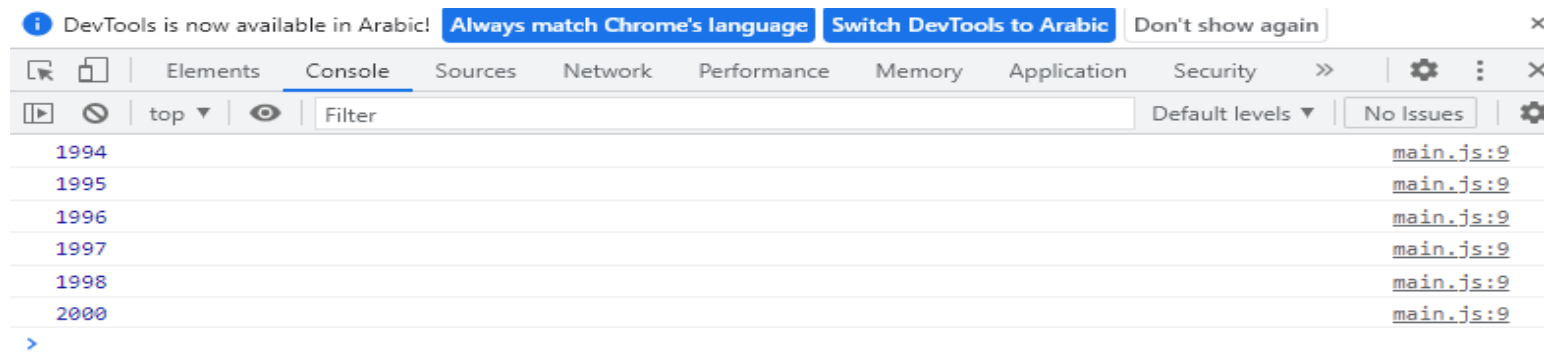
} test("ahmed" , 50)
test("hossam" , 30)
test("mohamed", 38) /* الارقام هي قيمة المتغير User */
test("ali", 25)
test("Hashem" , 18)
```



مثال استخدام الـ for و الـ If داخل الـ function

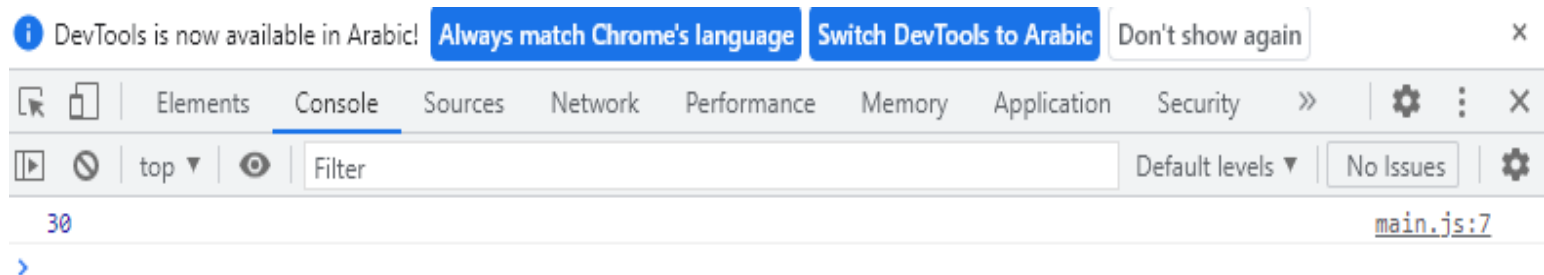
- يشرح المثال طباعة عدد من السنين باستثناء سنة معينة يتم تحديدها .

```
- function years(start , end , execlude){  
-     for(i=start;i<=end;i++){  
-         if(i===execlude){  
-             continue  
-         }  
-         console.log(i)  
-     }  
- } years(1994,2000,1999)
```



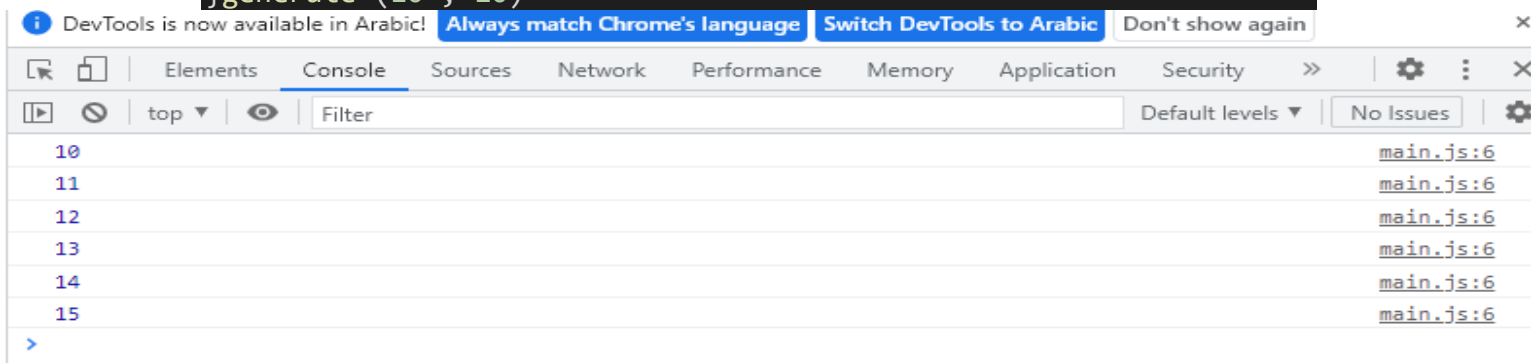
استعمال الـ return لارجاع القيمة

```
function names(num1,num2){  
    return num1+num2 //return لن يتم تنفيذ أى سطر برمجى بعد الـ  
}  
let result= names(20,10);  
console.log(result)
```



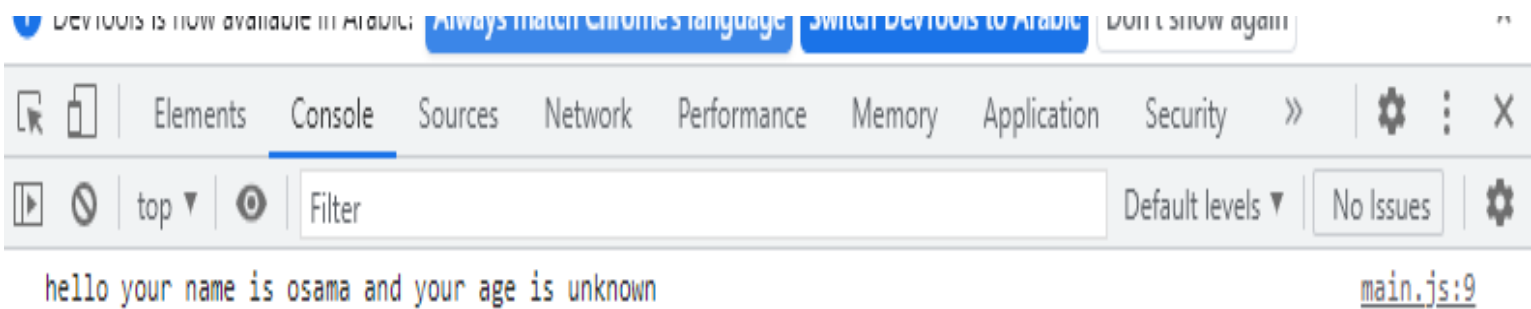
مثال اخر على الـ return

```
function generate (start , end){  
  
    for(i=start;i<=end;i++){  
        console.log(i)  
        if(i===15){  
            return `Interrupting` // تقوم بإيقاف عملية استمرار الدالة لتنفيذ الكود عند الوصول الى الرقم المحدد في الشرط  
        }  
    }  
}  
generate (10 , 20)
```



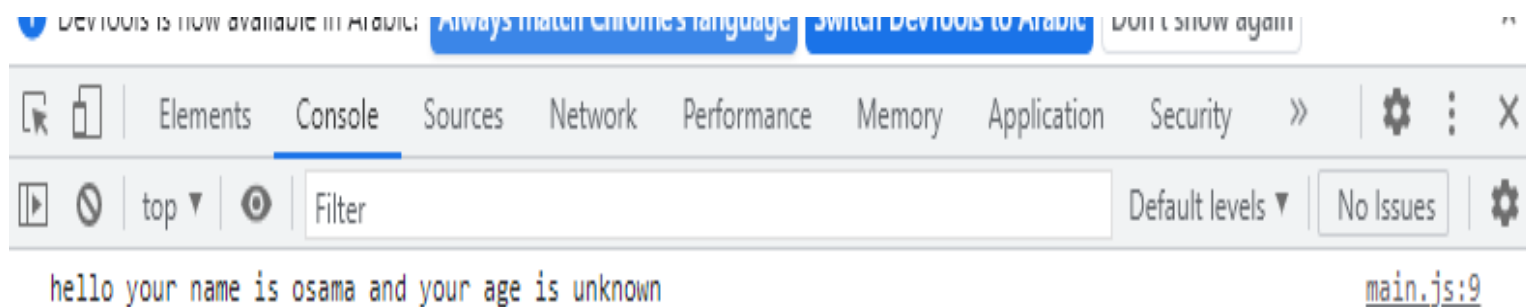
إذا كانت القيمة فارغة يكتب لك في المخرجات undefined وللتحكم في هذا الامر يمكن ان يتم وضع شرط له كالتالى :-

```
function generate(name , age){  
    if (age=== undefined){  
        age="unknown";  
    }  
    return `hello your name is ${name} and your age is ${age}`;  
}  
console.log(generate("osama"))
```



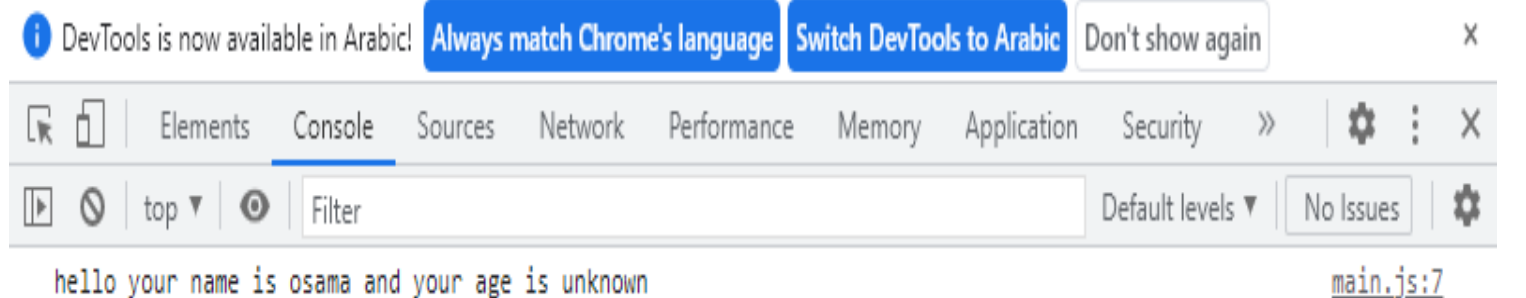
طريقة اخرى للتحكم فى العنصر غير المعرف undefined :-

```
function generate(name , age){
  age = age || "unknown"
  return `hello your name is ${name} and your age is ${age}`;
}
console.log(generate("osama"))
```



اسهل طريقة للتحكم فى العنصر غير المعرف undefined هي طريقة الـ **ecma**
او مايسمى بالـ **default value**

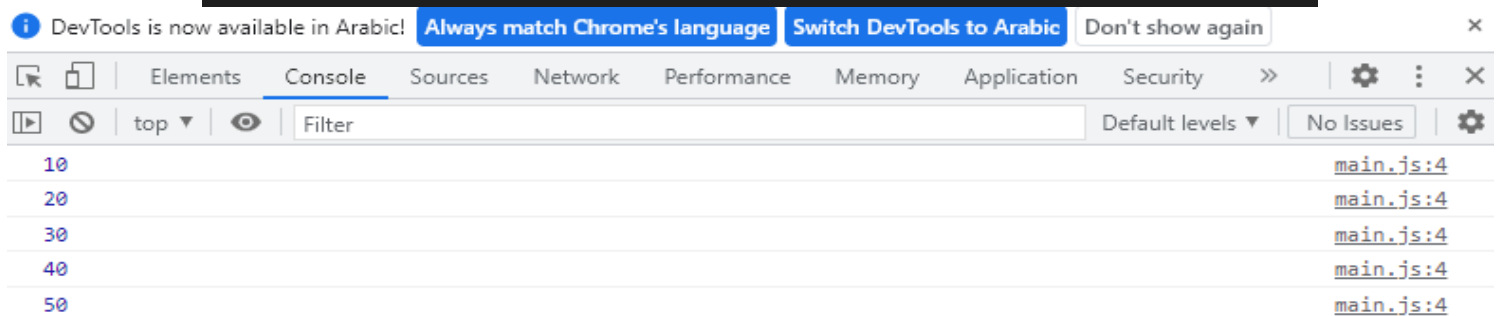
```
function generate(name , age="unknown"){// القيمة الافتراضية يمكن ان توضع لاي متغير
  return `hello your name is ${name} and your age is ${age}`;
}
console.log(generate("osama"))// يمكنك ايضا وضع قيمة لمتغير العمر وسيكون لها الاولوية فى المخرجات
```



شرح الـ rest parameter

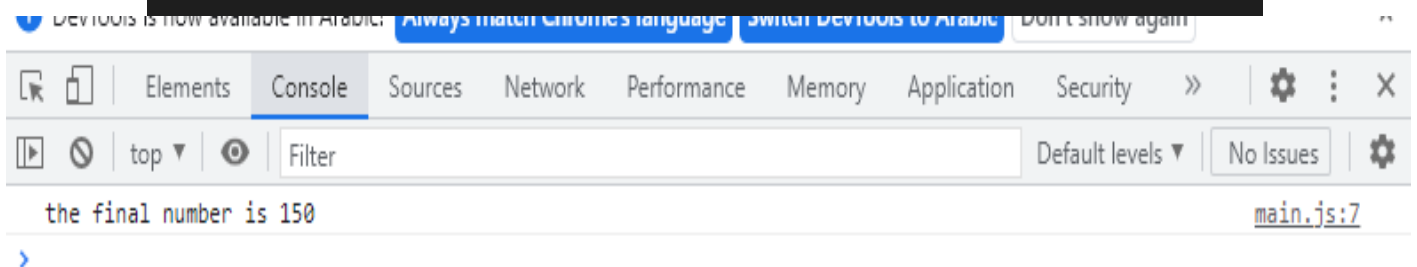
تسمح الـ function باستقبال عدد غير محدد من الـ arguments وعمل مصفوفة تحتوي على العديد من القيم ويرمز للـ rest parameter بالرمز (...variablename), مثال :-

```
function calc(...numbers){ // عمل مصفوفة
  for (i=0;i<numbers.length;i++){
    console.log(numbers[i])
  }
}calc (10 ,20 ,30 ,40 ,50) // numbers قيم المصفوفة
```



مثال على جمع عناصر المصفوفة لإخراج رقم واحد نهائي بطريقة الـ function :-

```
let result = 0;
function calc(...numbers){
  for (i=0;i<numbers.length;i++){
    result+=numbers[i] // result = result + number[i]
  }
  return `the final number is ${result}`
}console.log(calc (10 ,20 ,30 ,40 ,50) )
```



مثال متقدم على الـ function لظهور بيانات شخص ومهاراته :-

```

function showInfo(user="Un",age="Un",rt=0,show="yes",...sk){
document.write(`<div>`)
document.write(`<h1>Hello ${user}</h1>`)
document.write(`<p>your age is ${age}</p>`)
document.write(`<p>your watch price is ${rt}</p>`)
if(show==="yes"){ // اذا لم تتحقق تعود الى بديلها الاصلى
    if(sk.length>0){// اذا لم تتحقق تعود لبديلها الفرعى
        document.write(`<p>Skills: ${sk.join(" | ")}</p>`)
    } else { document.write(`<p>Skills: No Skills</p>`)// البديل
الفرعى
    }
} else { // البديل الاصلى للشرط
    document.write("Skills is Hiddien")
}
document.write(`</div>`)

}showInfo("Hossam" , 28,20 , "yes","hossam","islam")

```

Hello Hossam


your age is 28


your watch price is \$20

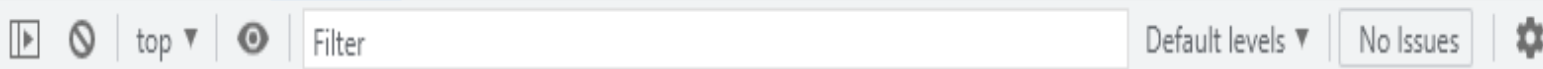
Skills: hossam | islam

مثال على عمل برنامج لادخال قيمة المتغير اذا كان من نوع نص فى المتغير name ورقم اذا كان من النوع رقم فى المتغير age وصح او خطأ اذا كان من نوع boolean فى المتغير status

```
function details (...array){
  let name ,age , staus, result;
  for(i=0;i<array.length;i++){
    if (typeof array[i]==="string"){
      name=array[i]
    }
    else if(typeof array[i]==="number"){
      age=array[i]
    }
    else if(typeof array[i]==="boolean"){
      staus=array[i]
      if(staus===true){
        result="you are avilable to hire"
      }
      else{
        result="you are not avilable to hire"
      }
    }
  }
  console.log(`name =>>>${name} || age=>>>${age} ||
status=>>>${staus} =>>> || result=>>> ${result}` )
} details(10 ,false, "hossam")
details("hossam",true, 30)
details(false ,10, "hossam")
```

 DevTools is now available in Arabic! [Always match Chrome's language](#) [Switch DevTools to Arabic](#) [Don't show again](#) ×





name =>>>hossam age=>>>10 status=>>>false =>>> result=>>> you are not avilable to hire	main.js:21
name =>>>hossam age=>>>30 status=>>>true =>>> result=>>> you are avilable to hire	main.js:21
name =>>>hossam age=>>>10 status=>>>false =>>> result=>>> you are not avilable to hire	main.js:21

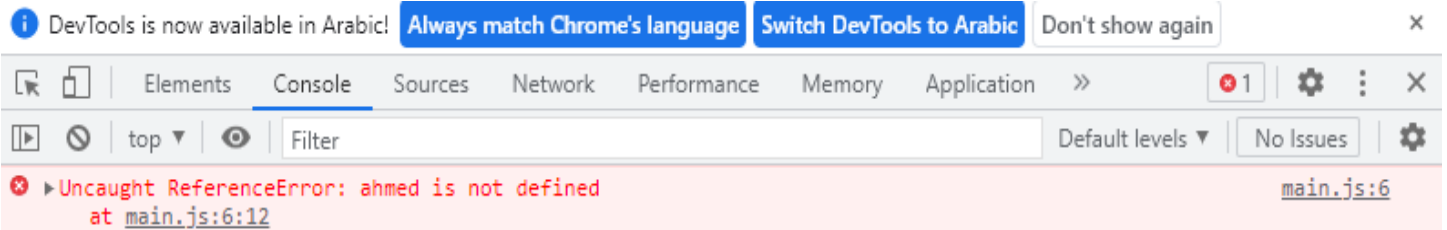
Anonymous Function

هي عبارة عن function بدون اسم كما في المثال التالي :-

```
let result=function (num1 , num2){  
  return num1+num2  
}; console.log(result(5,10))
```

لا يمكن كتابة جملة `console.log(result(10,5))` قبل انشاء الـ `function` لان الكود يقرأ من بداية انشاء المتغير الذي يحتوى على الدالة `function` واذا تم وضع اسم للـ `function` مع اسم المتغير سيظهر لك رسالة خطأ كما في الشكل التالي.

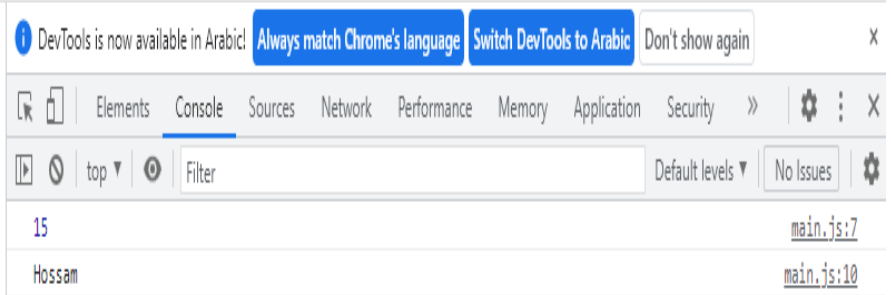
```
let result=function ahmed(num1 , num2){  
  return num1+num2  
}; console.log(ahmed(5,10))
```



يتم عدم الحاجة لتسمية الدالة واستخدام الطريقة **Anonymous Function** عند تنفيذ امر بسيط لا يحتاج الى تسمية الدالة كانشاء زر ينفذ لك امر عند الضغط عليه كما في المثال التالي:-

```
//html page  
<button id="show">show</button>  
//صفحة الجافا  
document.getElementById("show").onclick = function(){  
  console.log("Hossam");  
};
```

show



مثال على طباعة كلمة good فى الكونصول بعد ثانيتين:-

```
setTimeout(function(){// انشاء امر طباعة كلمة بعد ثانيتين من فتح الكونصول
  console.log("ahmed")
}, 2000);
```

ahmed

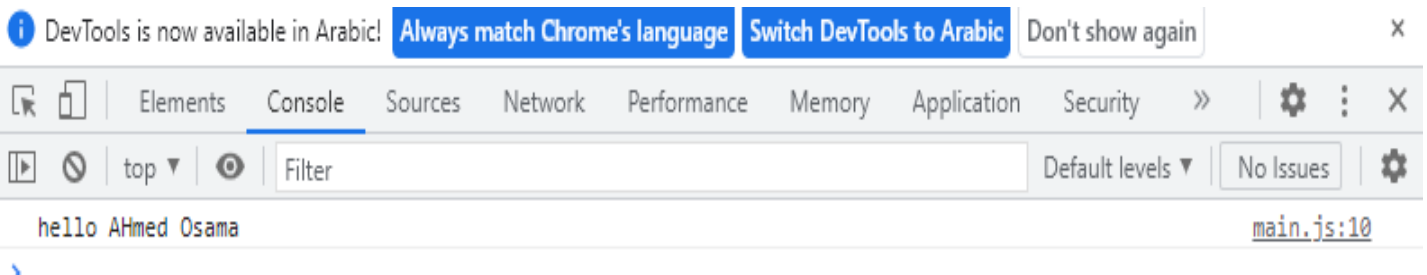
سواء قمت بتسمية الدالة او عدم تسميتها سيتم تنفيذ الكود بشكل صحيح ولا حاجة لتسميتها.

```
function sayHello (){
  console.log("Hello-World")
};
document.getElementById("show").onclick = sayHello; // انشاء دالة لها اسم وتنفيذها بمجرد الضغط على الزر
```

Nested Function

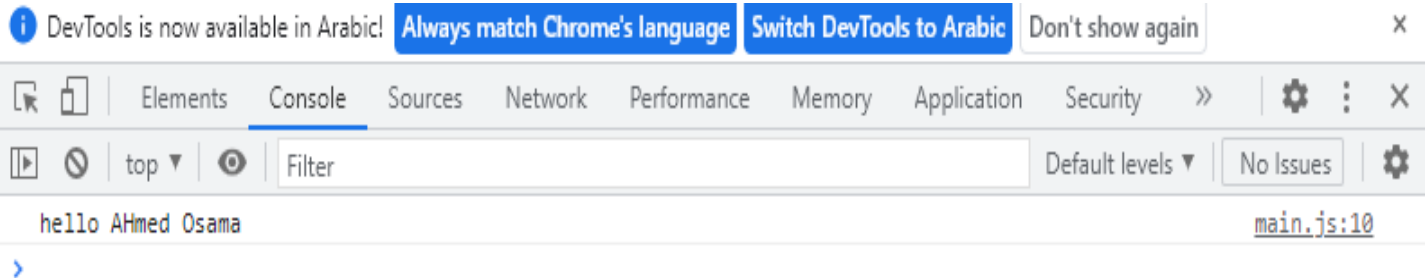
مثال 1 لدمج عناصر الـ function:-

```
function sayMessage(fName , lName){
  let message=`hello`
  function concatMessage (){
    message=`${message} ${fName} ${lName}` // message دالة لتحديث المتغير
  }
  concatMessage()
  return message; // ارجاع المتغير message بعد تحديثه فى الدالة الفرعية
}console.log(sayMessage("AHmed" , "Osama"))
```



مثال 2:-

```
function sayMessage(fName , lName){  
    let message=`hello`  
    function concatMessage (){  
        return `${message} ${fName} ${lName}` // ارجاع الثلاث متغيرات  
    } return concatMessage() // ارجاع الدالة الفرعية كقيمة  
}console.log(sayMessage("AHmed" , "Osama"))
```



مثال 3:-

مثال 3:-

```
function sayMessage(fName , lName){//الدالة الرئيسية
    let message=`hello`
    function concatMessage (){//دالة فرعية
```

```
JS main.js > sayMessage > concatMessage
1
2
3 function sayMessage(fName , lName){
4     let message=`hello`
5     function concatMessage (){
6         function generate(){
7             return `${fName} ${lName}`//ارجاع اصغر بقيمتين
8         }return `${message} ${generate()}`//ارجاع الدالة الفرعية بقيمة اصغر دالة
9     }
10    } return concatMessage() // ارجاع الدالة الفرعية كقيمة
11 }console.log(sayMessage("AHmed" , "Osama"))
12
13
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Code

[Running] node "c:\xampp\htdocs\Learn java script\main.js"
hello AHmed Osama

[Done] exited with code=0 in 0.217 seconds

[Running] node "c:\xampp\htdocs\Learn java script\main.js"
hello AHmed Osama

[Done] exited with code=0 in 0.266 seconds

```
function generate(){1 دالة فرعية
    function mm (){//2 دالة فرعية
        return `${fName}` //2 استرجاع الاسم الاول من الدالة الفرعية
    }
    return ` ${mm()} ${lName}` //1 استرجاع الدالة الفرعية 2 والاسم الاخير من الدالة الفرعية 1
}return `${message} ${generate()}`//استرجاع الدالة الفرعية 1 من الدالة الفرعية
} return concatMessage() //استرجاع الدالة الفرعية من الدالة الرئيسية
}console.log(sayMessage("AHmed" , "Osama"))
```

المعنى هنا من الامثلة الثلاثة انه يمكن استرجاع return للدالة الفرعية من الدالة الرئيسية.

Arrow Function

هي دالة تحفظ في متغير ويحذف منها كلمة فانكشن كما في المثال التالي :-

```
let show = ()=>{ // Arrow function يمكنك حذف كلمة
  فانكشن في
  return 10;
};
console.log(show());
```

[Done] exited with code=0 in 0.214 seconds

```
[Running] node "c:\xampp\htdocs\Learn java script\tempCodeRunnerFile.js"
10
```

يمكنك حذف الاقواس ايضا و الـ return في حالة وجود سطر واحد فقط في الدالة وسترجع لك نفس القيمة.

```
let show = ()=> 10;
console.log(show());
```

يمكنك ازالة الاقواس ايضا التي بجانب الرقم 10 ووضع _ مكانها .

```
let print = (num1 , num2)=>num1+num2 // اذا كان لديك عاملين
لا يمكنك ازالة الاقواس الموضوع فيها العامل
console.log(print(100,50))
```

يمكنك ازالة الاقواس فقط اذا كان هناك عامل واحد فقط.

Global and Local scope

امكانية الوصول الى المتغير من اى مكان طريق global scope والوصول الخاص بالدالة يكون من خلال الـ Local scope

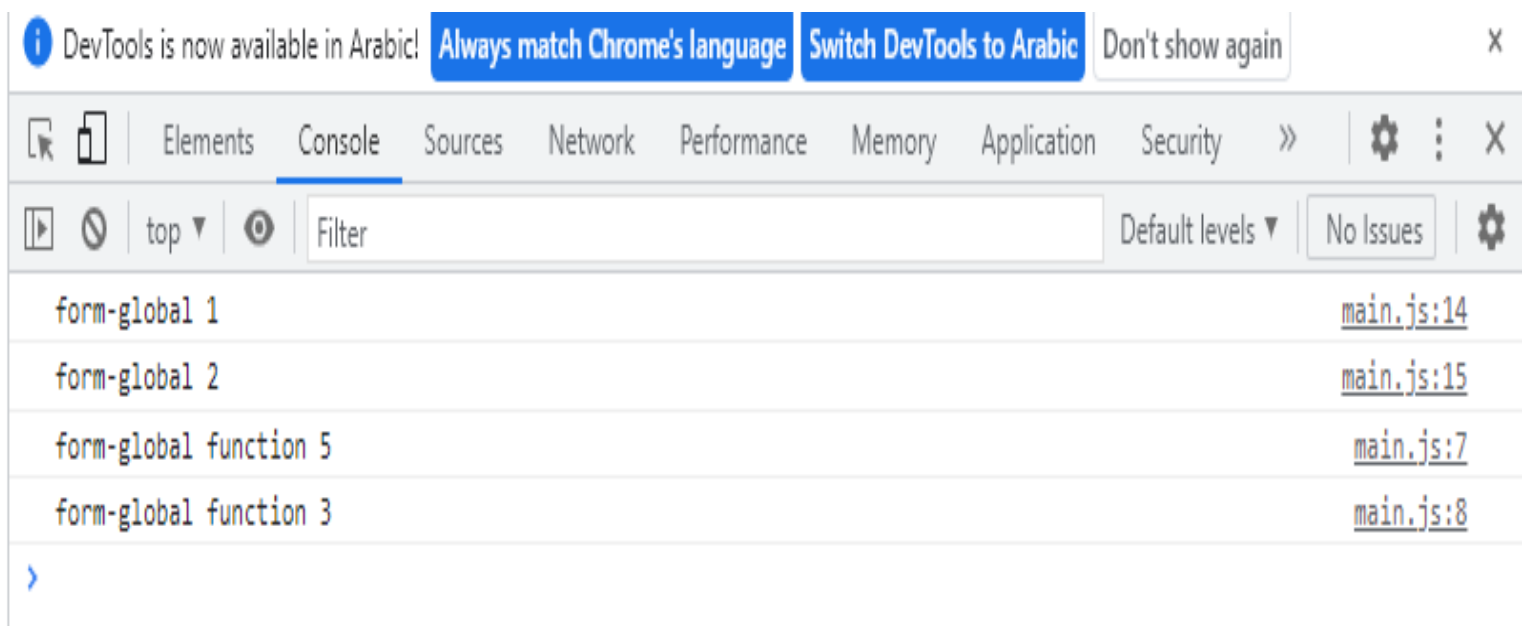
```
var a=1; // global scope — هنا النطاق العام الذى يمكن الوصول الى المتغير من اى مكان
let b =2;
function show (){
  var a=5; // local scope - متغيرات خاصة بالدالة فقط ولا علاقة لها بما خارج الدالة وعند
  | عدم وجودهم تبحث الدالة عن متغيراتها خارج نطاقه
  let b =3;
  console.log(`form-Local function ${a}`);
```



```

    console.log(`form-Local function ${b}`);
}
    console.log(`form-global ${a}`); // --- لن يستطيع الكونصول الخارج عن الدالة
    // --- Local Scope الخاص بالدالة Function يعني عند ازالة Global Scope
    // --- القراءة من النطاق الخاص بالدالة لا يمكن للكونصول القراءة من الـ
    console.log(`form-global ${b}`);
    show()

```



Block Scope

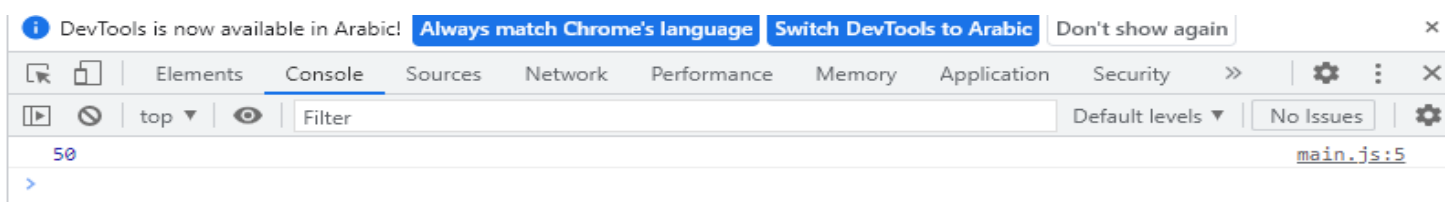
يوجد انواع لتعريف المتغير مثل **var** هذا النوع لا يشترط عليه شيء بمعنى انه يورث القيمة لاي دالة سواء كان الـ متغير داخل الدالة او خارجها . مثال :-

```

var x = 10;
if(10===10){
    x=50;
}
console.log(x)

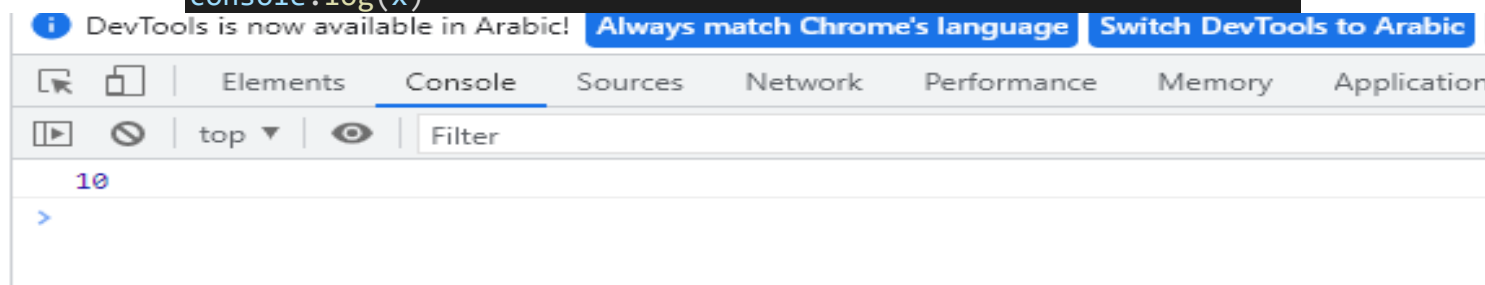
```

لاحظ في المخرجات ان المتغير ورث من الدالة If على الرغم انه ليس بداخلها.



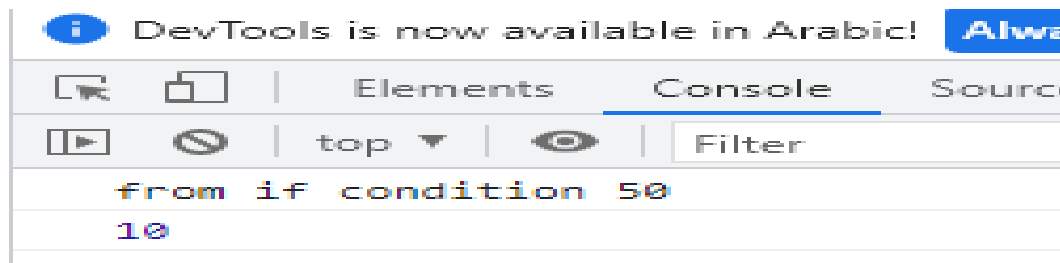
لاحظ نفس المثال ولكن باستخدام التعريف `let` للمتغير ستكون المخرجات من الـ `Global Scope`

```
let x = 10;
if(10===10){
  let x=50;
}
console.log(x)
```



لكن اذا كانت الامر كونيصل داخل الدالة سيرث المتغير `x` الذي بداخلها منها ولكن الذي خارج الدالة سيرث من الـ `GLOBAL scope`

```
var x = 10;
if(10===10){
  let x=50;
  console.log(`from if condition ${x}`)
}
console.log(x)
```



Lexi
cal
Scop

e

اذا كان هنا دالتين `2 functions` فان الدالة الفرعية `nested function` يمكنها ان ترث من متغيرات الدالة الاساسية ولكن لايمكن ان ترث الفرعية من الاساسية.

-يجب الوضع في الاعتبار ان الدالة تاخذ من المتغير الخاص بها اولا `Local Scope` وان لم يوجد تبحث في الـ `globale scope`

```
function parent (){
  let a = 10;
  function child(){
    let b = 20
    function grand(){
      let c = 30 ;
    }
  }
}
```

```

        console.log(b)// يمكن ان يرث مما هو اعلى منه
    }grand()
  }child()
}parent()

```

DevTools is now available in Arabic! [Always match Chrome's language](#) [Switch DevTools to Arabic](#) [Don't show](#)

Elements Console Sources Network Performance Memory Application Se

top Filter Defa

20

>

لاحظ انه تم طباعة المتغير لانه ورث من الدالة الاعلى منه والذي يتواجد بداخلها.

```

function parent (){
  let a = 10;
  function child(){
    let b =20
    console.log(c);// لايمكن ان يرث ممن هو اقل منه
    function grand(){
      let c = 30 ;
    }grand()
  }child()
}parent()

```

DevTools is now available in Arabic! [Always match Chrome's language](#) [Switch DevTools to Arabic](#) [Don't show a](#)

Elements Console Sources Network Performance Memory Application >>

top Filter Default leve

✖ ▶ Uncaught ReferenceError: c is not defined
 at child (main.js:5:21)
 at parent (main.js:9:6)
 at main.js:10:2

>

Hire Order Function

هى عبارة عن function تقبل function اخرى كعامل من العوامل وهذه الـ function التى يتم معاملاتها كعامل يكون لها ايضا عامل وتقوم بارجاع function :-

1. الـ Map:- هى عبارة عن ArrayMethod خاصة بالمصفوفات والتعامل معها فقط وانشاء مصفوفات جديدة.
2. تنشئ الـ Map مصفوفة جديدة ولا تتعامل مع المصفوفة الموجودة سابقا اى اذا تعاملت بها سننشئ لك مصفوفة جديدة ولن تطبق على المصفوفة الحالية.
 مثال :-

```

let myNums =[1,2,3,4,5,6];
let newArray=[];

```

```
for(i=0;i<myNums.length;i++){
    newArray.push(myNums[i]+myNums[i])
}console.log(newArray)
```

تطبيق نفس المثال بطريقة الـ Map :-

```
let myNums =[1,2,3,4,5,6];
let addSelf = myNums.map(function(element,index,arr){
    console.log(`current Element=> ${element}`)// العنصر الذى تتعامل معه
    console.log(`current Index=> ${index}`)// رقم العنصر الحالى الذى تتعامل معه
    console.log(`current Array=> ${arr}`)// المصفوفة الحالية التى تتعامل معها
    console.log(`This=> ${this}`) // الرقم المكتوب فى نهاية الدالة وهو الرقم
    10
},10);
```

DevTools is now available in Arabic! [Always match Chrome's language](#) [Switch DevTools to Arabic](#) [Don't show again](#)

Elements Console Sources Network Performance Memory Application Security >> | Settings

top Filter Default levels No Issues

▶ (6) [2, 4, 6, 8, 10, 12]	main.js:5
current Element=> 1	main.js:9
current Index=> 0	main.js:10
current Array=> 1,2,3,4,5,6	main.js:11
This=> 10	main.js:12
current Element=> 2	main.js:9
current Index=> 1	main.js:10
current Array=> 1,2,3,4,5,6	main.js:11
This=> 10	main.js:12
current Element=> 3	main.js:9
current Index=> 2	main.js:10
current Array=> 1,2,3,4,5,6	main.js:11
This=> 10	main.js:12
current Element=> 4	main.js:9
current Index=> 3	main.js:10
current Array=> 1,2,3,4,5,6	main.js:11
This=> 10	main.js:12
current Element=> 5	main.js:9
current Index=> 4	main.js:10
current Array=> 1,2,3,4,5,6	main.js:11
This=> 10	main.js:12
current Element=> 6	main.js:9
current Index=> 5	main.js:10
current Array=> 1,2,3,4,5,6	main.js:11
This=> 10	main.js:12

لاحظ فى المخرجات انه يعرض لك كل من العنصر element ورقمه والمصفوفة التى تتعامل معها كاملة

تطبيق المثال لجمع عناصر المصفوفة :-

```
let myNums =[1,2,3,4,5,6];
let addSelf = myNums.map(function(element,index,arr){
    return element + element
},5);console.log(addSelf);
```

▶ (6) [2, 4, 6, 8, 10, 12]

>

استعمال نفس المثلث بالـ -:Arrow Function

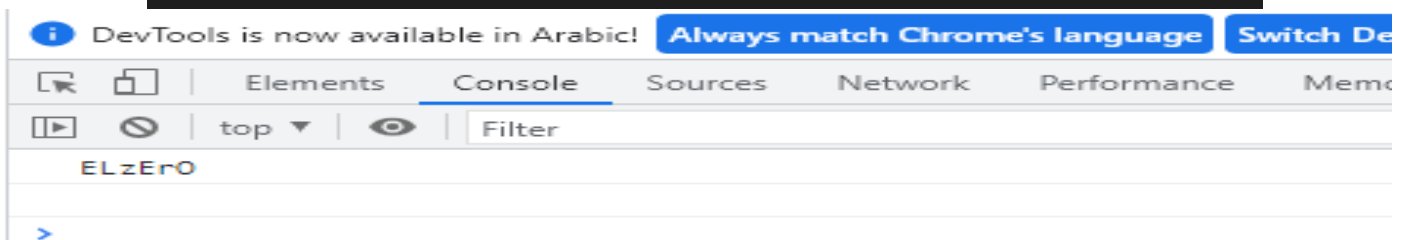
```
let addSelf = myNums.map((element)=>element +
element );
console.log(addSelf);
```

استعمال نفس المثلث بالـ Additional function التي تقوم بجمع العنصر على نفسه.

مثال على تحويل قيمة متغير الى مصفوفة بالدالة split وتحويل الحروف الصغيرة الى كبيرة والعكس باستخدام الـ .map

```
let swaaping="elZeRo"
let invertedNumbers=[1,-10,-20,15,100,30]
let ignoreBoolean="Elz123er4o"

let sw=swaaping.split("").map(function(ele){ // بتحويل قيمة المتغير الى
تقوم split مصفوفة ليتم التعامل معها
//condition ? true : false ---- convert capital letters to
small letters and convert small letters to capital letters
return /*if*/ ele === ele.toUpperCase()?/*if true=>>*/
ele.toLowerCase() : /*else*/ele.toUpperCase() /*if false */
}).join("")
console.log(sw)
```



```
/* تحويل عناصر المصفوفة السالب الى موجب والعكس */
let invertedNumbers=[1,-10,-20,15,100,30]
let inv =invertedNumbers.map(function(element){
return -element;
});
console.log(inv);
```

► (6) [-1, 10, 20, -15, -100, -30]

>

مثال اخر لتصفية الحروف من الارقام فى الكلمة:-

إذا كان العنصر رقم قم بعمل تخطى له اما اذا كان نص فقم بارجاعه :

```
let ignoreBoolean="Elz123er4o"
let ignore=ignoreBoolean.split("").map(function(element){
return isNaN(element)? element : "" ;
}).join("")
console.log(ignore);
```

Elzero

>

تحويل الامثلة الثلاثة الى طريقة Arrow Function

```
let swaaping="elZeRo"
let invertedNumbers=[1,-10,-20,15,100,30]
let ignoreBoolean="Elz123er4o"
```

```
let sw=swaaping.split("").map((ele)=> ele === ele.toUpperCase()?
ele.toLowerCase() :ele.toUpperCase());console.log(sw.join(""))
```

```
let inv =invertedNumbers.map((element)=> -
element);console.log(inv);
```

```
let ignore=ignoreBoolean.split("").map((element)=>isNaN(element)?
element : "" );console.log(ignore.join(""));
```

Filter

يقوم بارجاع عناصر المصفوفة التى تم اختبارها , على سبيل المثال اذا تم عمل IF Condition على مصفوفة فان العناصر التى حققت قيمة true هى التى سيرجعها الـ filter وهو يطبق نفس خصائص الـ map ولكن الفرق بينهم ان الـ map ترجع قيمة العملية سواء كانت جمع او ضرب او قسمة ولكن اذا كان ناتج الدالة true او false تقوم الـ map بارجاعها كما هى. ولكن الـ Filter اذا وجدت قيمة المتغير true تقوم بارجاعه كما هو وسيوضح ذلك فى المثال التالى:-

```
let array = [10,20,30,40,50]
```

```

let add = array.map(function(element){
    return element + element
}); console.log(add)

let fil = array.filter(function(element){
    return element + element
}); console.log(fil)

```



لاحظ هنا ان الـ map قامت بجمع قيم المصفوفة اما الـ filter فقامت بارجاعها كما هي عندما وجدت قيمتها تساوى true.

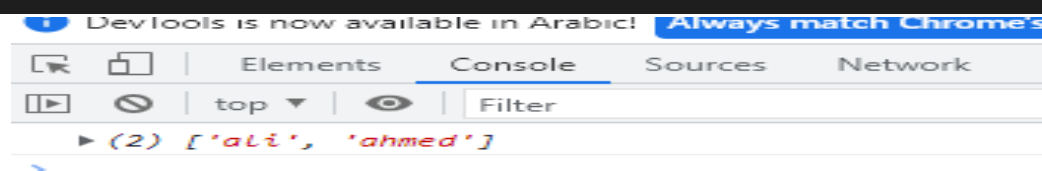
مثال 1 :-

قم بعمل فلتره للاسماء التى تبدأ بحرف الـ a من المصفوفة.

```

let friends =
["ali","ahmed","hossam","hazem","omar","odai","fars","fawzy"]
let numbers =[11,20,2,5,17,10]
let array = friends.filter(function (element){
    return element.startsWith("a"?true : false
});console.log(array);

```



مثال على تصفية الارقام الزوجية من المصفوفة والتي نعرفها من ناتج باقى القسمة على 2 يساوى صفر.

```

let evenNumbers = numbers.filter(function (element){

```

```
return element % 2 ===0 // 2 على القسمة على 2
يساوى صفر
});console.log(evenNumbers);
```

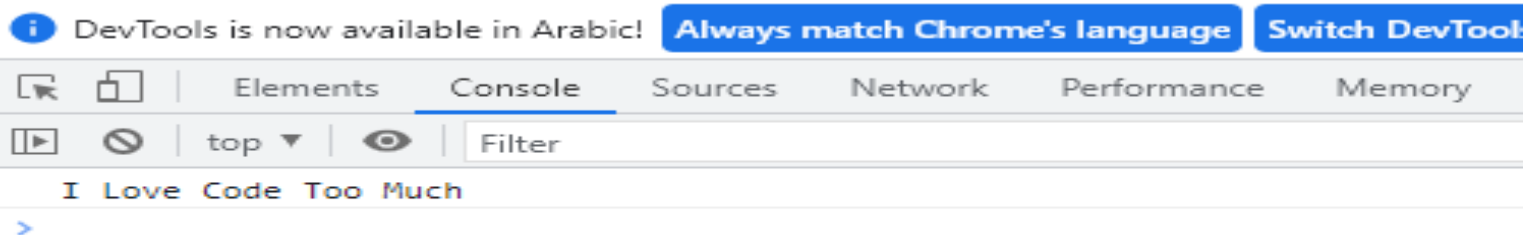
▶ (3) [20, 2, 10]

>

مثال على فلتر الكلمات التي يزيد عدد حروفها عن 5 حروف.

```
let sentence = "I Love Food Code Too Playing Much"

let sen = sentence.split(" ").filter(function(element){
    return element.length<=4 // ارجاع الكلمات التي لا يزيد عدد حروفها عن 4 حروف
});console.log(sen.join(" "));
```



مثال على تصفية الارقام من كلمة وضربها باستعمال map وال filter مع بعض.

- ولكن تذكر ان ال map تقوم بالعمليات الحسابية وترجع لكن ناتجها بينما ال filter تقوم بارجاع العنصر كما هو دون اجراء عملية حسابية عليه,,, وتقوم ال map بارجاع العنصر كما هو تماما في حالة كانت المخرجات true او false ولكن ال filter تقوم بتصفية العناصر وارجاعها لك بعد التصفية ولا ترجع true او false. اي انك يجب ان تفكر اولاً ما اذا كنت ستجرى عملية حسابية او ستقوم بتصفية العناصر .

```
let mix = "A13BS2ZX"
let multiply =
mix.split("").filter(function(element){//Filter تصفية
العناصر اولاً في دالة ال
return isNaN(parseInt(element))?"": element
}).map(function(element){ //Map اجراء العمليات الحسابية
ثانياً في دالة ال
return element*element
});console.log(multiply)
```

▶ (3) [1, 9, 4]

>

Reduce

تقوم بتنفيذ الـ function على كل عنصر من عناصر المصفوفة كما هو الحال في الـ map والـ filter ولكنها تقوم بإرجاع أقل قيمة يمكن اختصارها وتقليصها.

تقبل الـ reduce الـ callback function والـ Initial value .

- يوجد في الدالة 4 عناصر وهي (arr , index , current , acc) :-

acc:-فرضا إذا أردنا إرجاع مجموع عناصر المصفوفة فإن الـ acc هو حاصل الجمع الذي سيتم جمعه مرة أخرى مع العنصر الجديد.

Current:- هو العنصر الجديد الذي سيتم جمعه مع حاصل الجمع وهو الـ acc.

Index:-رقم العنصر ,,,, arr:-المصفوفة التي تتعامل معها.

Initial value:- هي القيمة التي تبدأ بها عملية جمع المصفوفة أي هي قيمة acc التي تحددها أنت وتكتب في آخر الدالة كما في المثال التالي.

مثال:-

```
let nums = [10,20,30,40,50]
let add = nums.reduce(function(acc , current ,index,arr) {
  console.log(acc);
  console.log(current);
  console.log(index);
  console.log(arr);
  console.log(acc+current);
  console.log("#####");
  return acc+current;
},5//=>>Initial Value is 5);console.log(add);
```

-----لم يتم طباعة المخرجات نظرا لكبر حجم الصورة-----

مثال على تصفية حروف من مصفوفة ولصقها ببعض لتعطيك كلمة باستخدام دالة **filter** و دالة **reduce**

```
let removechars = ["h","@","o","@","s","@","s","@","a","m",]
let remoove=removechars.filter(function(element){
return !element.startsWith("@")
}).reduce(function(acc,current){
return `${acc} ${current}`
})
;console.log(remoove)
```

```
h o s s a m
>
```

ForEach

تطبق الدالة **forEach** على كل عنصر من عناصر المصفوفة دون انشاء مصفوفة جديدة ويفضل استعمالها عند عدم الحاجة الى تغيير بيانات المصفوفة.

مثال :-يشرح المثال التالي اضافة قائمة **ul** وبمجرد الضغط على **li** داخلها يتم حذف **class** يسمى **active** من جميع **li** الموجودة واطافة **class** يسمى **active** في العنصر الذي تم النقر عليه فقط.

```
<ul class="ul">
  <li class="active">one</li>
  <li>two</li>
  <li>three</li>

</ul>
<div class="content">
  <div>div_one</div>
  <div>div_tow</div>
  <div>div_three</div>
</div>
```

```
let allis = document.querySelectorAll("ul li")
let allDivs= document.querySelectorAll(".content div")
```

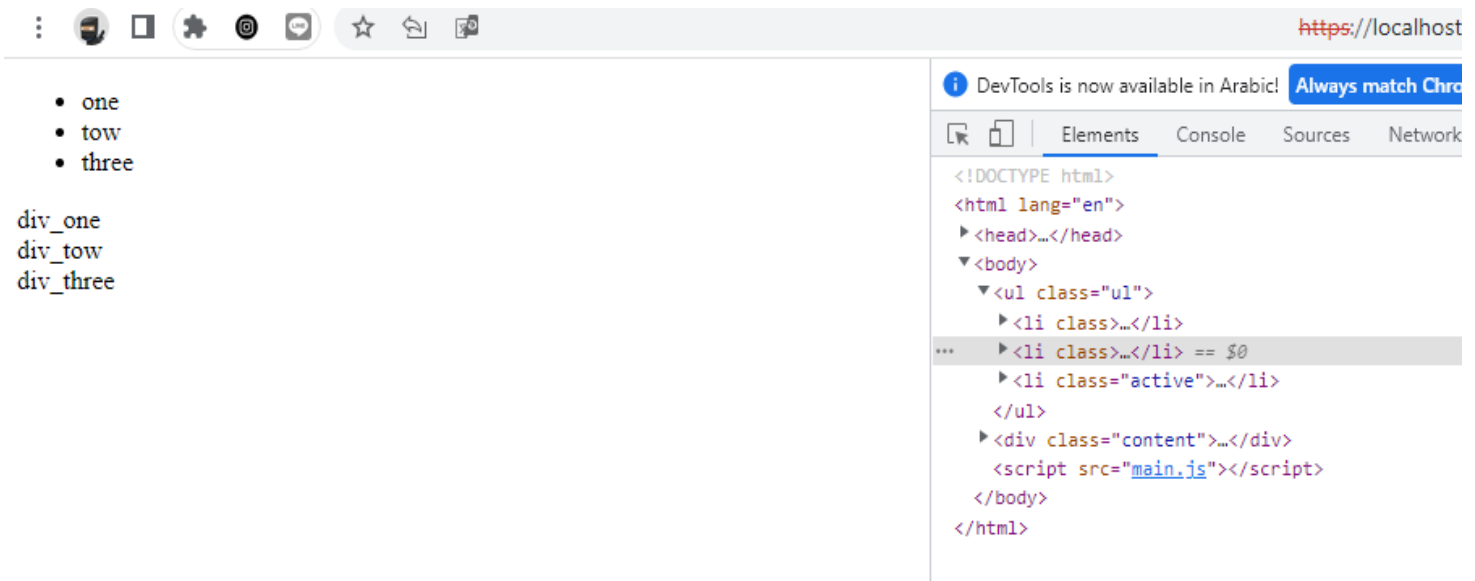
```

allis.forEach(function(element){
    element.onclick=function(){
        allis.forEach(function(element){
            element.classList.remove("active") //remove class
active from all li elements
        })
    this.classList.add("active") // add classe named active in ech
li you press on
    allDivs.forEach(function(element){ //ul  عناصر من عناصر


element.style.display="none";
    })
})
});


```

ملحوظة :- تم انشاء فانكشن داخل النقر على العنصر onclick لازالة المطلوب ولم يتم انشاء function داخل onclick لاضافة المطلوب لأن عملية الحذف تتطلب عمل loop على جميع العناصر لاجراء عملية الازالة في جميع العناصر اما عملية الاضافة فتتم في عنصر واحد فقط وليس في جميع العناصر.



حل التحدي :-

```
let mystring
="1,2,3,EE,l,z,e,r,o,_,W,e,b,_,S,c,h,o,o,l,2,0,z"

let solution = mystring.split(",").map(function(element){

return element.includes("_")?"":
isNaN(element)?element:""
}).filter(function(element){
return element.startsWith("")? element : ""
})
.reduce(function(acc,current){
return acc+current
})
;console.log(solution.slice(1,7) , solution.slice(7,10) ,
solution.slice(10,16)
)
```

What is Object

يحتوى الـ object على خصائص properties و methods:-

ماهى الـ properties:- هى خصائص تقوم بارجاع بيانات مثل:-

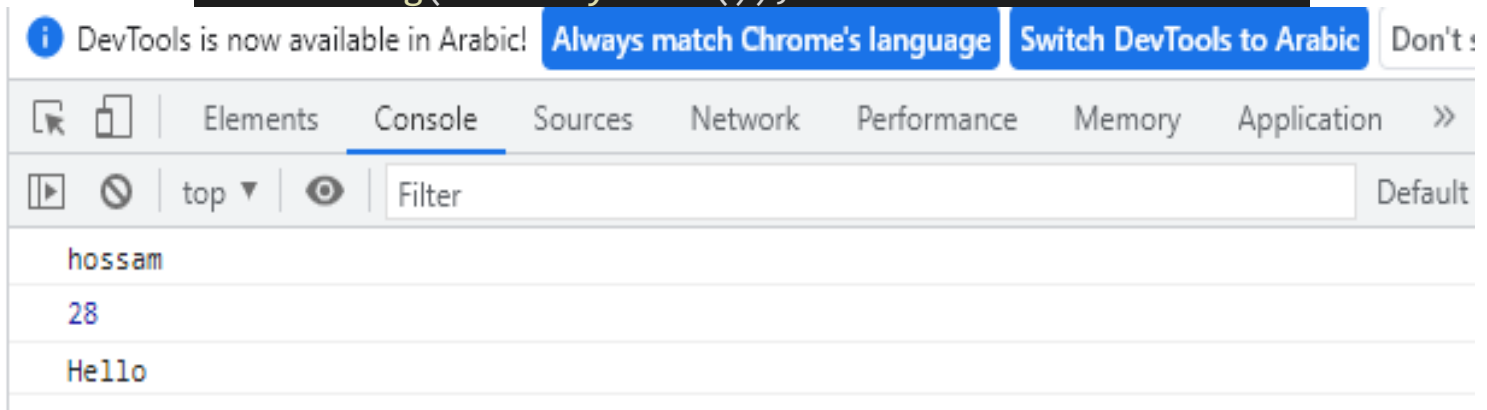
Window Type:- هى عبارة عن properties of object من ضمن الخصائص التى ترجع معلومات

window.location.href:- تعرض عنوان الموقع وهى nested object يرجع لك بيانات من نوع string.

Window.location.assign(<https://google.com>):- ليست properties مثل الـ عنصرين السابقين الذان يقومان بارجاع معلومات وانما هى method تقوم بتغيير عنوان الموقع.

طريقة كتابة الـ object:-

```
let user = { // object طريقة كتابة الـ
  // properties of object خصائص الـ
  theName: "hossam",
  age : 28 ,
  // Methods
  sayHello : function(){
    return "Hello";
  },
};
// Object للوصول الى ما بداخل الـ
console.log(user.theName);
console.log(user.age);
console.log(user.sayHello());
```



دور الـ bracket notation في الوصول الى خصائص الـ object و طباعتها حيث يمكن من خلالها عمل متغير داخل الكائن بصيغة نصية والوصول الى قيمة هذا المتغير و طباعته من خلال الـ bracket notation .

```
let user = {
  theName: "hossam",
  "the country" : "is egypt" // صيغة نصية تعمل كمتغير له قيمة
};
console.log(user["the country"]) // bracket notation access
```



top ▼



Filter

is egypt



يمكن ايضا استخدام الـ bracket notation للوصول الى قيمة المتغير حتى لو كان المتغير ليس موضوعا داخل علامة تنصيب ولكن لابد من وضع علامة التنصيب للمتغير داخل الـ bracket notation للمتغير كما في المثال التالي :-

```
let user = {
  theName: "hossam",
};
console.log(user["theName"]) // bracket notation يمكن وضع المتغير دون
نقطة تسبقه بطريقة الـ
```

- يمكن عمل متغير خارج الـ object يحتوي على قيمة ,, هذه القيمة هي عبارة عن متغير داخل الـ object وهذا المتغير يحتوي على قيمة وعند طباعة المتغير المتواجد خارج الـ object سيقوم بارجاع قيمة القيمة الموجودة داخل الـ object مثال :-

```
let myVar = "name"; // قيمة متغير خارج الكائن تعمل كمتغير له
قيمة داخل الكائن

let user = {
  name : "Hossam" , //myVar او name يتم الوصول الى
  هذه القيمة من خلال طباعة الـ
};
console.log(user[myVar]) // طباعة قيمة المتغير خارج
الكائن تكون غير مسبقة بنقطة ولا تحتوي على علامة تنصيب
console.log(user.name)
console.log(user["name"])
```



top ▼



Filter

Hossam

Hossam

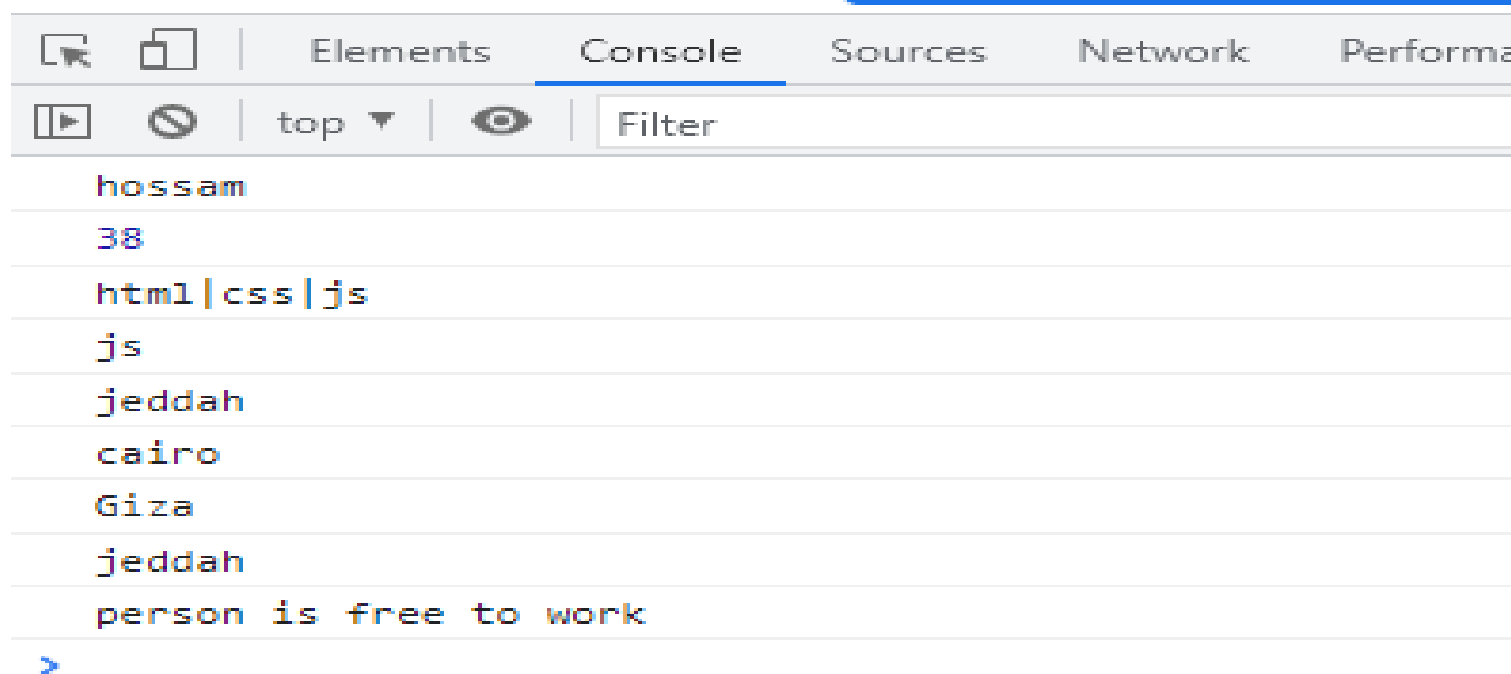
Hossam



Nested Object

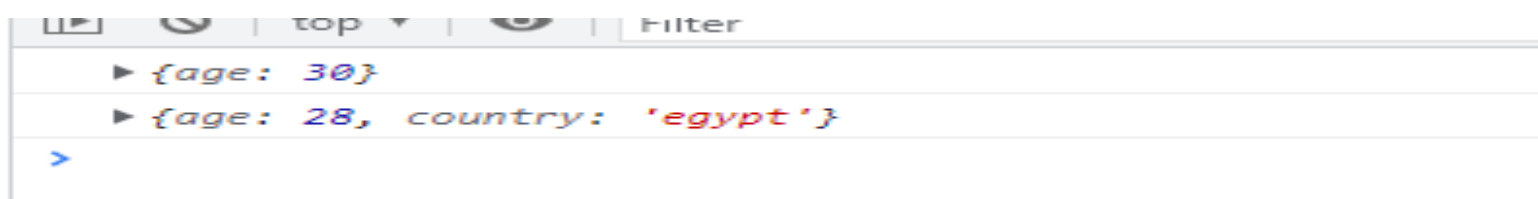
مثال على إنشاء object داخل object و إنشاء method داخل الـ object الرئيسي.

```
let available = true;
let user = {
  name : "hossam",
  age : 38,
  skills : ["html" , "css" , "js"],
  available : true,
  /*first nested object */
  addresses : { // nested object contain more than property
    ksa : "jeddah",
    /*second nested object*/
    egypt : {
      one : "cairo",
      tow : "Giza"
    },
  },
  /*method in object */
  checkPerson : function(){ //create methode to check if
the employer is available to work or not
if(user.available===true){
return "person is free to work"
}else{
return "person is not available to work"
}
}
};
console.log(user.name);
console.log(user.age);
console.log(user.skills.join("|"));
console.log(user.skills[2]); // access to array element with index
console.log(user.addresses.ksa); //access to nested object property
console.log(user.addresses.egypt.one); // you can't access to
nested nested object by writing ksa.egypt.one
console.log(user["addresses"]["egypt"]["tow"]); // access by
bracket notation way
console.log(user["addresses"].ksa); //you can also use 1 bracket
notation
console.log(user.checkPerson()); //
```

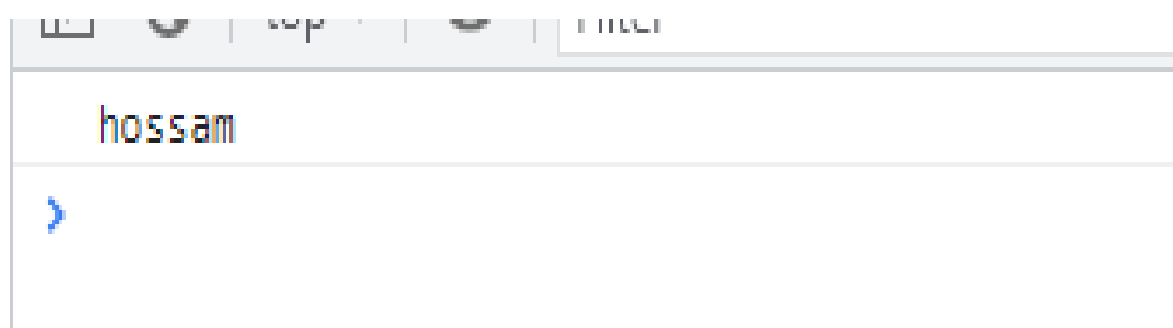


طريقة جديدة لإنشاء الـ object

```
let user = {  
  age:30, // العمر قبل التحديث  
};  
console.log(user);  
/* يمكن كتابة خصائص الكائن خارج أقواس الكائن بالطريقة التالية */  
user.age = 28; // العمر بعد التحديث  
user["country"] = "egypt";  
console.log(user);
```



```
let user = {}  
user.check=function() {  
  return `hossam`  
}  
console.log(user.check())
```



طريقة انشاء object بطريقة الـ new Keyword

```
let user = new Object();

user.age= 28;
user["country"] = "egypt"
user.check=function(){
    return `hello`
}
console.log(user.age)
console.log(user.country)
console.log(user.check())
```

28

egypt

hello

>

```
let user = new Object({
    age:20,
});
console.log(user.age)
user.age= 28;
user["country"] = "egypt"
user.check=function(){
    return `hello`
}
console.log(user.age)
console.log(user.country)
console.log(user.check())
```

20

28

egypt

hello

>

This Key Word

This:- يتم تعيينها للكائن الذي يتم استدعائها عليه حيث ان المثال التالي يشرح عملية طباعة **this** التي ستقوم بارجاع **window** وهو العارضة التي نستعرض فيها اوامر الكود وكذلك في السطر البرمجي الثاني تم عمل اختبار فيه يسأل ما اذا كانت **this** هي الـ **window** الذي يتم استدعاء الامر **this** عليه ام لا :-

```
console.log(this);  
console.log(this===window); // هل يقصد الكود اننا الامر يريد استدعاء الويندو
```

```
Window {window: Window, self: Window, document: document, name: '', Location: Location, ...} main.js:1  
true main.js:3  
---
```

مثال اخر بطريقة الـ **function**:-

```
function sayHello(){//this عمل دالة تستدعي الامر  
    return this;  
} sayHello()  
  
console.log(sayHello())  
console.log(sayHello()===window) // هل يقصد بها الويندو
```

DevTools is now available in Arabic! Always match Chrome's language Switch DevTools to Arabic Don't show again

Elements Console Sources Network Performance Memory Application >> 1 Issue: 1

```
Window {window: Window, self: Window, document: document, name: '', Location: Location, ...} main.js:8  
true main.js:9  
>
```

مثال اخر يتم فيه انشاء **button** ليتم انقر عليه وعند النقر عليه يظهر لنا ان ما قمنا بالنقر عليه هو الـ **button** الذي انشأته:-

```
document.getElementById("cl").onclick = function(){  
    console.log(this); //button هنا على المالك وهو الزرار تعود This  
};
```

Click

https://localhost:10443/Learn%20java%20script/index.html غير آمن

DevTools is now available in Arabic! Always match Chrome's language Switch DevTools to Arabic Don't show again

Elements Console Sources Network Performance Memory Application >> 1 Issue: 1

```
<button id="cl">Click</button> main.js:6
```

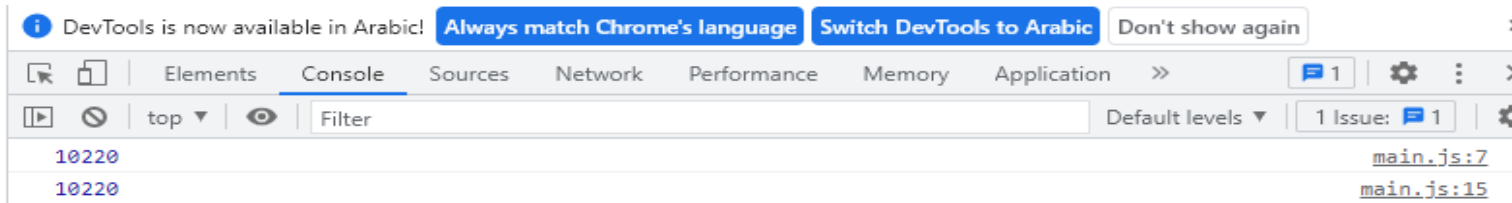
يتم الحاجة الى استخدام `this` لتحديد الشيء الذى قمنا بالنقر عليه لظهار خصائص محددة فى هذا الشيء نريد معرفتها.

- مثال اخر يوضح فيه استخدام `this` داخل دالة انشائها داخل `object`:

```
let user = {
  age : 28,
  ageIn_Days:function(){
    return user.age*365
  },
};
console.log(user.ageIn_Days());

let user1 = {
  age : 28,
  ageIn_Days:function(){
    return this.age*365 //user الذى تم انشائه وهو الـ object
    //هنا على الـ object الذى تم انشائه وهو الـ user
    //يعود الامر الى this
  },
};
console.log(user1.ageIn_Days());
```

لاحظ فى المخرجات ان الامر `this` قام بارجاع نفس النتيجة التى تم فيها استخدام اسم الكائن `user` لاجراء عملية الضرب داخل الـ `function`.



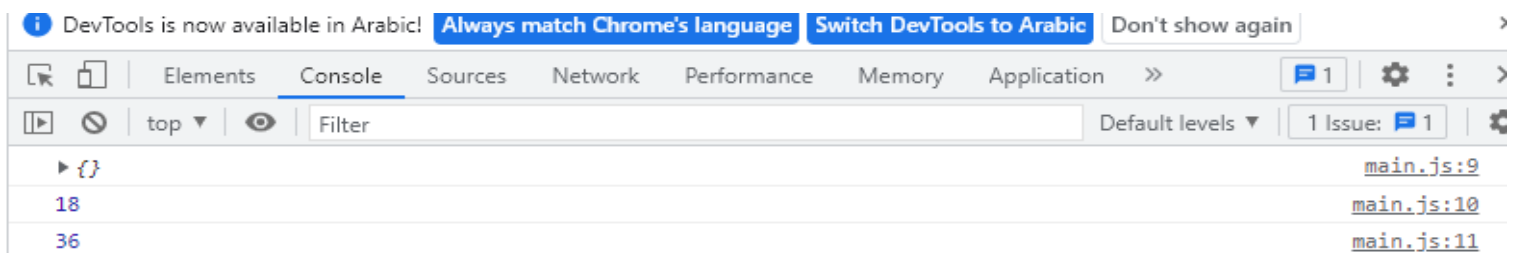
شرح طريقة انشاء كائن `object` باستعمال طريقة الـ `create`

يمكن من خلال `object.create` استدعاء كائن داخل الكائن الذى يحتوى على `object.create` والتعامل مع محتويات هذا الكائن الذى تم استدعائه كأنها محتويات الكائن `object.create` وسيوضح ذلك المثال التالى :-

```
let user = { // هنا تم انشاء الكائن
  age : 28,
  doubleDays:function(){
    return user.age*2
  },
};
let cobObject = Object.create (user) // هنا تم استدعاء الكائن
console.log(cobObject)
console.log(cobObject.age)
```

```
console.log(cobObject.doubleDays())
```

لاحظ في المخرجات انه قد تم اجراء العمليات التي تم انشائها في الكائن الذي تم استدعائه.

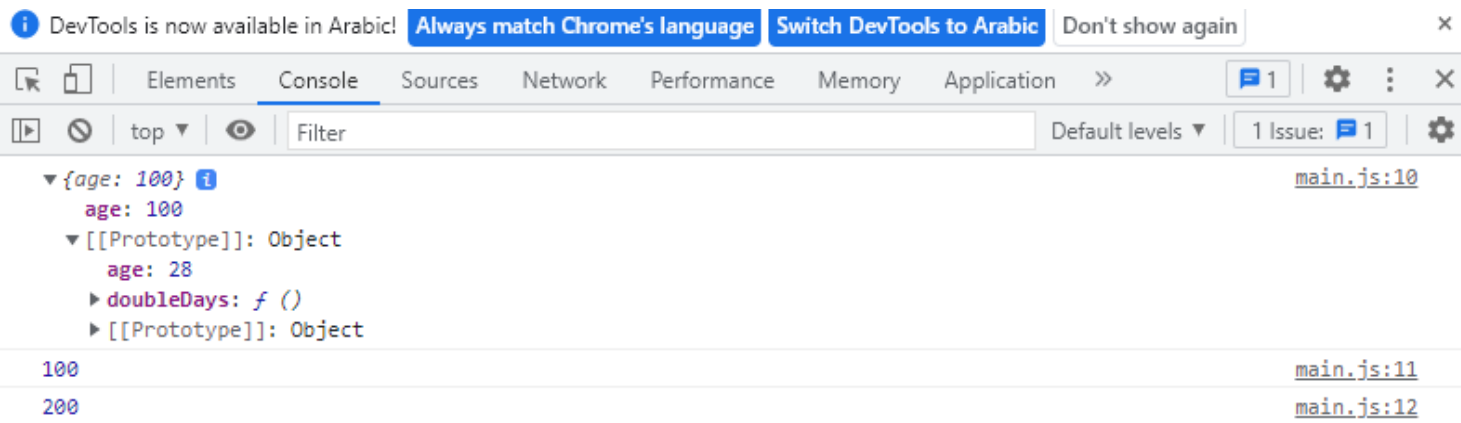


```
let user = { // هنا تم انشاء الكائن
  age : 28,
  doubleDays:function(){
    return user.age*2
  },
};
let cobObject = Object.create (user) // هنا تم استدعاء الكائن
cobObject.age = 30;
console.log(cobObject.doubleDays())
```

في المثال السابق تم تحديث العمر بعد انشاء الكائن ولكن عند تنفيذ الكود لطباعة الدالة doubleDays فانه لن يرجع لك age الذي تم تحديثه في الكائن user cobObject مضروباً في 2 ولكن سيرجع لك age الموجود في الكائن user. doubleDays الدالة ترجع user.age وليس cobObject.age ولحل هذه المشكلة تكتب this.age بدلا من user.age حيث ان this تعود على الكائن الذي يستدعي الدالة وهو الكائن cobObject الذي قام باستدعاء الكائن user بمحتوياته كما في المثال التالي.

```
let user = {
  age : 28,
  doubleDays:function(){
    return this.age*2 // this تعود على الكائن الذي يستدعي الدالة
  },
};
let cobObject = Object.create (user) doubleDays الدالة
cobObject.age = 100;

console.log(cobObject)
console.log(cobObject.age)
console.log(cobObject.doubleDays())
```



شرح استعمال الـ assign مع الـ object

تقوم الـ assign ب جلب معلومات اكثر من كائن لاضافتها فى كائن اخر .

```
let finalObject = Object.assign(targetObject,obj1);
```

فى السطر البرمجى السابق يمثل `obj1, targetObject` الكائنات التى سيتم جلب معلوماتها لاضافتها فى كائن اخر.

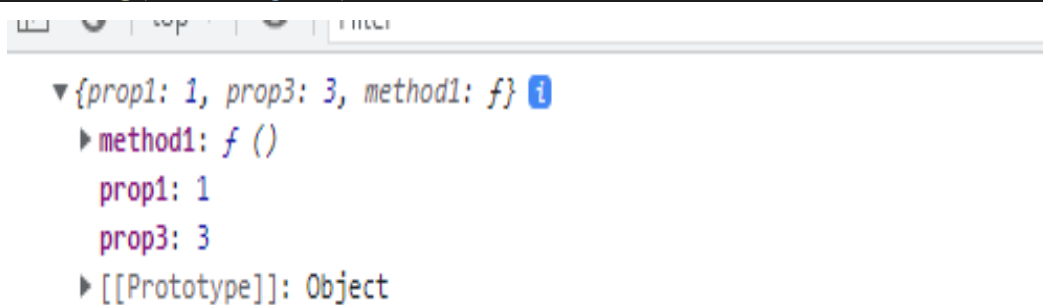
اذا تم جلب معلومات اكثر من كائن وكان هناك متغيران بنفس الاسم فى كائنين مختلفين ي فان الكائن الذى يستضيف معلومات الكائنين سيقوم باخذ قيمة اول كائن كما فى المثال التالى :-

```
let obj1={
  prop1 : 1,
  method1 : function(){
    return this.prop1;
  },
};

let obj2 = {
  prop2 : 2 ,
  method2 : function() {
    return this.prop2;
  },
};

let targetObject = {
  prop1 : 100, // لن يتم اعتبارها فى الكائن المستضيف لان يوجد متغير بنفس الاسم فى الكائن
  السابق
  prop3: 3 ,
}

let finalObject = Object.assign(targetObject,obj1);
console.log(finalObject);
```

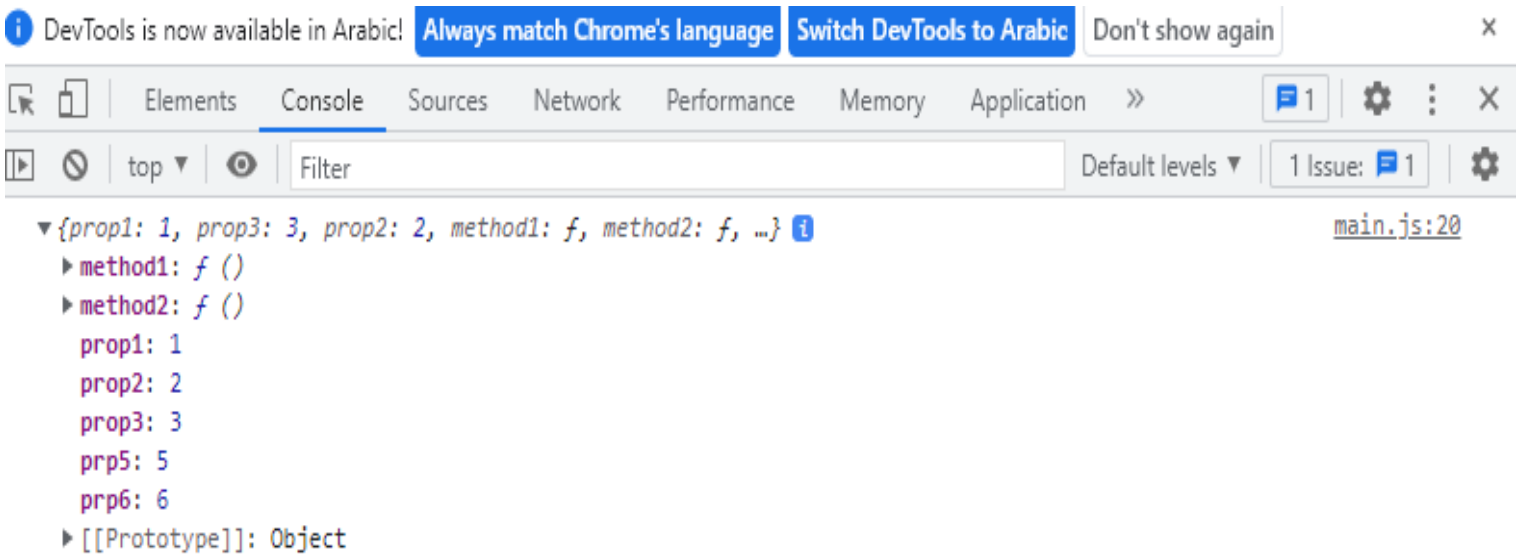


يمكن ايضا اضافة الكائن نفسه الذى قام باستضافة كائنات اخرى فيه عن طريق الـ assign كما فى المثال التالى :-

```
let obj1={
  prop1 : 1,
  method1 : function(){
    return this.prop1;
  },
};

let obj2 = {
  prop2 : 2 ,
  method2 : function() {
return this.prop2;
  },
};

let targetObject = {
  prop1 : 100, // لن يتم اعتبارها فى الكائن المستضيف لان يوجد متغير بنفس الاسم فى الكائن
السابق
  prop3: 3 ,
}
let finalObject = Object.assign(targetObject,obj1,obj2);
let newObject = Object.assign({},finalObject,obj1 ,
{prp5:5,prp6:6} ) // كائن يستضيف الكائن السابق الذى قام باستضافة كائنات اخرى مع انشاء
متغيرات جديدة
console.log(newObject)
```



Dom

ماهو ال Dom -: Document Object Model عندما يحدث للصفحة load
يتم انشاء model للصفحة من خلال المتصفح هذا ال model هو عبارة عن
object يحتوى على جميع العناصر التى تكون من خلالها قادر على التعامل مع جميع
عناصر الصفحة.

- مثال على الوصول الى بعض عناصر الصفحة html بالاي دى و اسم التاج
-:tag name

```
- </head>
- <body>
-   <span class="my-span">My Span</span>
-   <p>Hello Paragraf 1</p>
-   <p>Hello Paragraf 2</p>
-   <div id="my-div">Hello Div</div>
-   <form action="">
-       <input type="text" name="one" value="Hello"/>
-   </form>
-   <form action="">
-       <input type="text" name="two" value="Hello"/>
-   </form>
-   <a href="https://google.com">Google</a>
-   <a href="https://facebook.com">Facebook</a>
-   <script src="main.js"></script>
- </body>
- </html>

- /*كود الجافا*/
- الوصول للعنصر عن طريق الID
- let myIdElement = document.getElementById("my-div");
- console.log(myIdElement);

- الوصول الى العناصر وليس عنصر واحد عن طريق اسم العنصر Tag
- let myTagElement = document.getElementsByTagName("p");//
- console.log(myTagElement);
- الوصول الى عنصر واحد باسم التاج
- console.log(myTagElement[1])//
- تغيير العنوان بداخل البرجراف الثانى
- myTagElement[1].innerHTML="Test" ;
```

استخدام الـ `querySelector` للوصول الى العناصر

تتيح لك الوصول الى العناصر سواء باستخدام اسم الكلاس او الـ `id` او اسم التاج ولكنها لاتجلب الا عنصر واحد فقط اذا كان هنا اكثر من عنصر على عكس `.getelementBy`

```
/*by querySelector you can access to element by id or by class name or by taG NAME */

let myQueryselectorClassName = document.querySelector(".my-span");//access by class name
let myQueryselectorID = document.querySelector("#my-div");//ACCESS by id
let myQueryselectorTagName= document.querySelector("p");// access by tag name

console.log(myQueryselectorClassName);

console.log(myQueryselectorID);

console.log(myQueryselectorTagName);
```

My Span My Span 2

Hello Paragraf 1

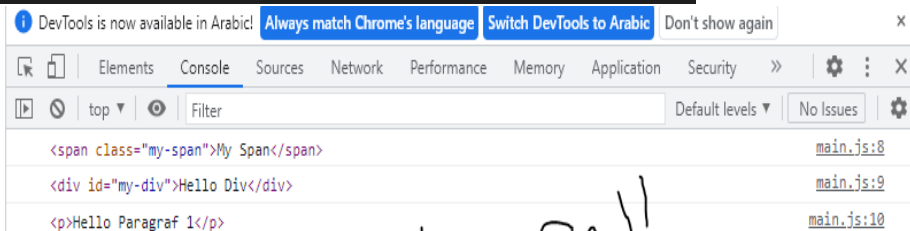
Hello Paragraf 2

Hello Div

Hello

Hello

[Google](#) [Facebook](#)



الوصول الى عنصر واحد

`querySelectorAll` تتيح لك الوصول الى جميع العناصر سواء باستخدام ID او استخدام TagName او استخدام الـ `className` :-

```
let myQueryselectorClassName = document.querySelectorAll(".my-span");//access by class name
let myQueryselectorID = document.querySelectorAll("#my-div");//ACCESS by id
let myQueryselectorTagName= document.querySelectorAll("p");// access by tag name
```



```

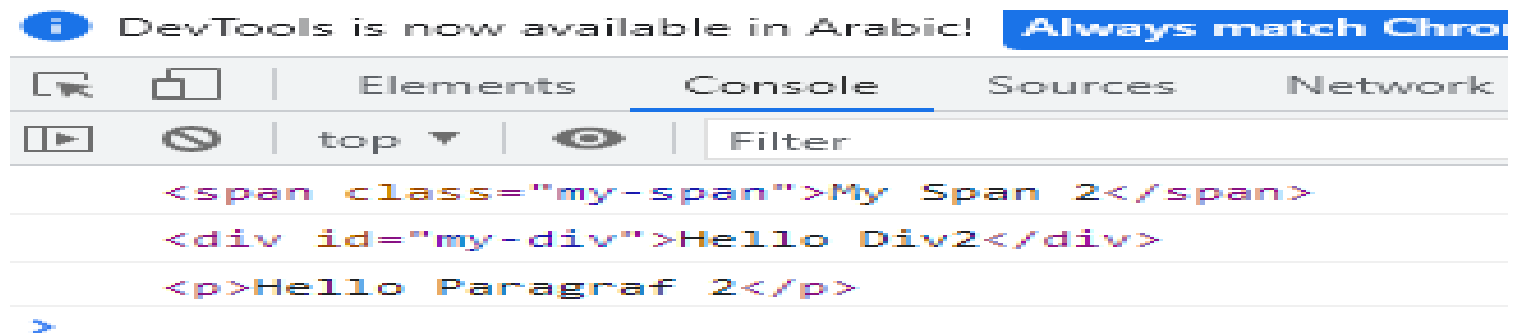
console.log(myQueryselectorClassName[1]); //access to second span

console.log(myQueryselectorID[1]); //access to second Div

console.log(myQueryselectorTagName[1]); //access to second
paragraf

```

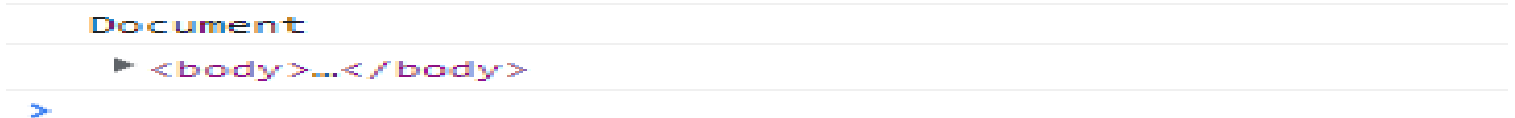
المثال السابق يوضح الوصول الى العنصر الثانى باستخدام `querySelectorAll`



```

console.log(document.title); // الوصول الى عنوان الصفحة لاجراء التغييرات التى تريدها
console.log(document.body); // الوصول الى هيكل الصفحة لعمل التغييرات التى تريدها

```



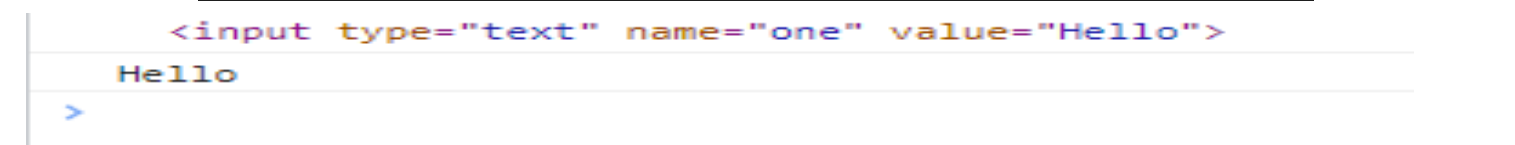
الوصول الى الـ `forms` من خلال الكونصول عن طريق تحديد الاندكس الخاص به والاسم والقيمة

```

<form action="">
  <input type="text" name="one" value="Hello"/>
</form>
<form action="">
<input type="text" name="tow" value="Hello"/>
</form>

.....
console.log(document.forms[0].one); // access to form 1 by it's
name
console.log(document.forms[0].one.value); // you can also access
to form value

```



الوصول الى الروابط الموجودة في الصفحة باستخدام الكونصول:-

```
<a href="https://google.com">Google</a>
  <a href="https://facebook.com">Facebook</a>

.....

console.log(document.links[0])
console.log(document.links[1])
```

```
<a href="https://google.com">Google</a>
<a href="https://facebook.com">Facebook</a>
```

```
console.log(document.links[1].href)//access to link value
```

```
https://facebook.com/
```

الان كيفية الوصول الى محتوى العناصر والتعديل عليها .

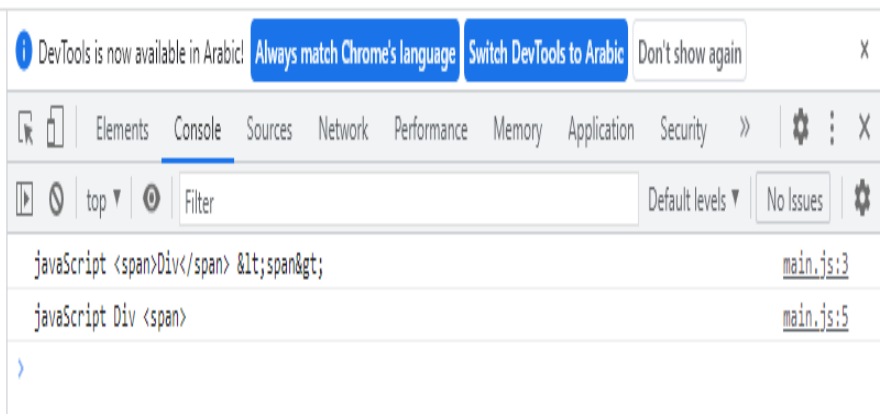
```
&lt;span>
```

يكتب النص البرمجي السابق لاطها كلمة span في منتصف عنوان الصفحة
على انه تاج Tag

```
let myElement = document.querySelector(".js");

console.log(myElement.innerHTML);// يرجع كود الـ HTML زي ما هو
console.log(myElement.textContent);// يرجع المحتوى النصي الموجود في العنصر
```

JavaScript Div
[Google](#)



```
myElement.innerHTML = "Text from <span>Js</span> " // لتغيير كود الـ HTML
```



Text from Js
[Google](#)

```
myElement.textContent = "Text from <span>Js</span> " //Html تغيير
محتوى نص ال
```



Text from Js
[Google](#)

فى الصورة السابقة يعتبر الكود الـ span على انه text وليس عنصر
 html

```
/*html code*/
```

```
<img src="" alt=""/>
```

```
document.images[0].src = "https://google.com" //تغيير الصورة
```

```
document.images[0].alt = "Alternate" //alt تغيير عنصر ال
```



Alternate [Google](#)

```
document.images[0].title = "Hossam"// اضافة صفة او شئ ناقص للعنصر حيث اننا لم
نكتب صفة العنوان فى الكود
```

اضافة id ايضا والـ classname :-

```
document.images[0].id = "pic";
document.images[0].className = "img";
```

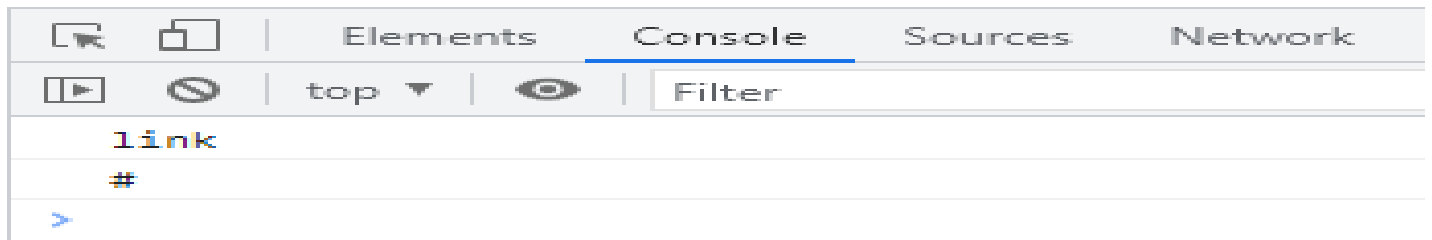
```
<div class="js">Text from <span>Js</span> </div>
```

```
 == $0
```

```
<a href="#">Google</a>
```

لجلب صفة من الصفات مثلا اذا اردنا معرفة اسم الـ class او عنوان الـ href:-

```
<a class="link" href="#">Google</a>
/* جلب خصائص عنصر */
myAttribute = document.querySelector(".link")
console.log(myAttribute.getAttribute("class"))
console.log(myAttribute.getAttribute("href"))
```



الان بعد ماتم طباعة خصائص العنصر داخل الكونصول يمكننا التعديل عليه كالتالى :-

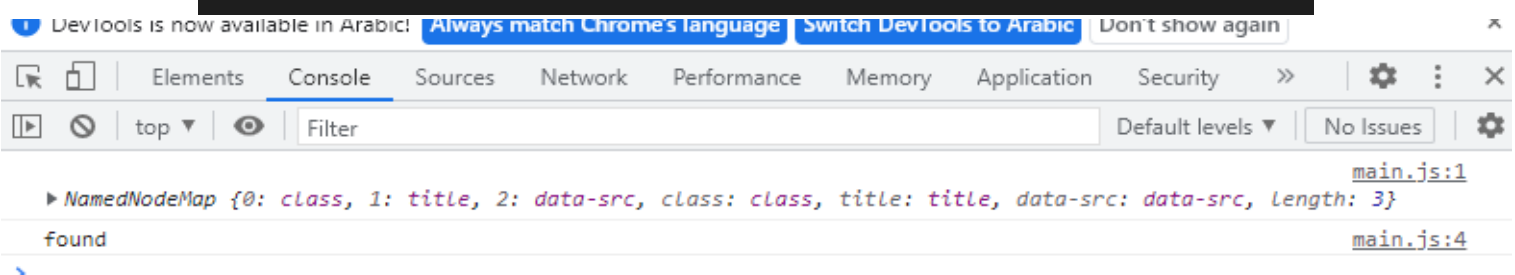
```
<a class="link" href="#">Google</a>
.....
// تغيير صفة العنصر حيث يكتب في الخانة الاولى الصفة والخانة الثانية يكتب قيمة الصفة التي نريد تغييرها
myAttribute.setAttribute("href" , "https://facebook.com")
//facebook اضافة صفة الـ title والتي قيمتها تكون
myAttribute.setAttribute("title" , "facebook")
```

hasAttribute

هل العنصر لديه هذه الصفة ام لا كما في المثال التالي :-

```
<div > Div </div>
<p class="para" title="paragraf" data-
src="Testing">paragraf</p>
<img src="" alt="" />
```

```
.....
console.log(document.getElementsByTagName("p")[0].attributes);
let myp = document.getElementsByTagName("p")[0];
if(myp.hasAttribute("data-src")){
    console.log("found")
}else{
    console.log(`not found`)
}
```



مثال على حذف صفة باستخدام removeAttribute

```
console.log(document.getElementsByTagName("p")[0].attributes);
let myp = document.getElementsByTagName("p")[0];
if(myp.hasAttribute("data-src")){
myp.removeAttribute("data-src");//حذف صفة
}else{
    console.log(`not found`)
}
```

```
▼ <body>
  <div> Div </div>
  <p class="para" title="paragraf">paragraf</p>
  <img src(unknown) alt>
  <script src="main.js"></script>
</body>
</html>
```

لاحظ في الـ (P) انه قد تم حذف صفة "data-src"

مثال اخر على اختبار صفة العنصر بحيث اذا كانت قيمتها فارغة سيتم حذفها واذا كانت ليست فارغة سيتم تعيين قيمة جديدة لها.

كود Html

```
<div > Div </div>
  <p class="para" title="paragraf" data-
src="Testing">paragraf</p>
  <img src="" alt="" />
```

كود الجافا

```
console.log(document.getElementsByTagName("p")[0].attributes);
let myp = document.getElementsByTagName("p")[0];
if(myp.hasAttribute("data-src")){
    if(myp.getAttribute("data-src")===""){
        myp.removeAttribute("data-src")
    }else{
        myp.setAttribute("data-src" , "new value");
    }
}else{
    console.log(`not found`)
}
```

لاحظ في المخرجات انه قد تم تعيين قيمة جديدة للصفة "data-src" حيث ان قيمتها لم تكن فارغة.

```
<body>
<div> Div </div>
<p class="para" title="paragraf" data-src="new value">paragraf</p>
<img src(unknown) alt>
<script src="main.js"></script>
</body>
```

مثال على اختبار هل العنصر له صفات ام لا :-

في المثال التالي عنصر الـ Div لا يوجد له صفات او خصائص تميزه.

```
<div > Div </div>
```

الان اجراء عملية الاختبار:-

```
if(document.getElementsByTagName("div")[0].hasAttributes()){
    console("element has attributes")
}else{
    console.log("element has no attributes")
}
```

المخرجات :-

```
element has no attributes
```

```
>
```

createElement

تمكنك من انشاء جميع محتويات الصفحة خطوة بخطوة حتى لو كانت تخص صفحة الـ html:-

لانشاء عنصر الـ div:-

```
let myElement = document.createElement("div")
console.log(myElement)
```



top ▼



Filter

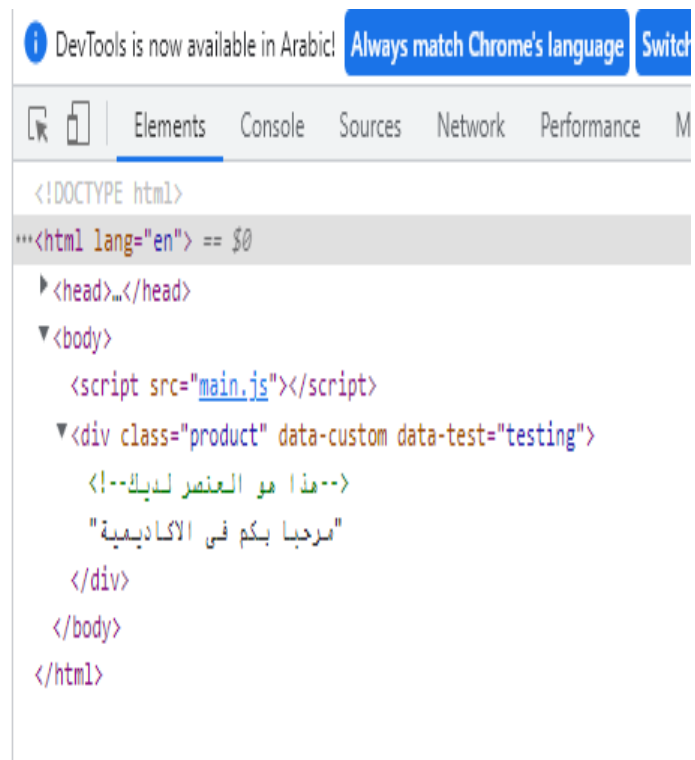
```
<div></div>
```

```
>
```

الان لانشاء خصائص العنصر ووضعها داخل العنصر:-

```
// انشاء العنصر من البداية
let myElement = document.createElement("div")
// انشاء صفة من تلقاء نفسك غير معروفة يمكنك تغييرها في المستقبل
let myAttribute = document.createAttribute("data-custom")
// لانشاء النص المكتوب داخل العنصر
let myText = document.createTextNode("مرحبا بكم في الاكاديمية")
// لانشاء تعليق
let comment = document.createComment("هذا هو العنصر لديك")
// انشاء صفة موجودة او معروفة للعنصر
myElement.className="product"
// ربط الخاصية او الصفة التي انشأتها بالعنصر
myElement.setAttributeNode(myAttribute)
// انشاء صفة غير معروفة ووضع قيمة لها/
myElement.setAttribute("data-test", "testing")
// وضع التعليق قبل النص
myElement.appendChild(comment)
// جلب المتغير الذي يحتوى على النص داخل العنصر
myElement.appendChild(myText)
// وضع العنصر الذي انشأته داخل الصفحة وليس داخل الكونصول
document.body.appendChild(myElement)
```

مرحبا بكم في الاكاديمية



مثال متقدم على انشاء العناصر وخصائصها عن طريق انشاء منتج يحتوى على خصائص :-

```
// انشاء العنصر
let mainElement = document.createElement("div");
let headElement = document.createElement("h2")
let paragraphElement = document.createElement("p")

let headText = document.createTextNode("مرحبا بكم فى منتجات الزيرو")
let paragraphText = document.createTextNode("لدينا افضل المنتجات على الاطلاق فى الشرق الاوسط")

// create class name to div element
mainElement.className="main"
// put headtext inside headElement
headElement.appendChild(headText)
//put paragraph text inside pagrafElement
paragraphElement.appendChild(paragraphText)
//put head element inside div element
mainElement.appendChild(headElement)
//put paragraph element inside div element
mainElement.appendChild(paragraphElement)
document.body.appendChild(mainElement)
```

مرحبا بكم فى منتجات الزيرو

لدينا افضل المنتجات على الاطلاق فى الشرق الاوسط

DevTools is now available in Arabic! [Always match Chrome's language](#) [Switch DevTools to Arabic](#) [Don't show again](#)

Elements Console Sources Network Performance Memory Application Security »

```
<!DOCTYPE html>
<<<html lang="en"> == $0
  <head>...</head>
  <body>
    <script src="main.js"></script>
    <div class="main">
      <h2>مرحبا بكم فى منتجات الزيرو</h2>
      <p>لدينا افضل المنتجات على الاطلاق فى الشرق الاوسط</p>
    </div>
  </body>
</html>
```

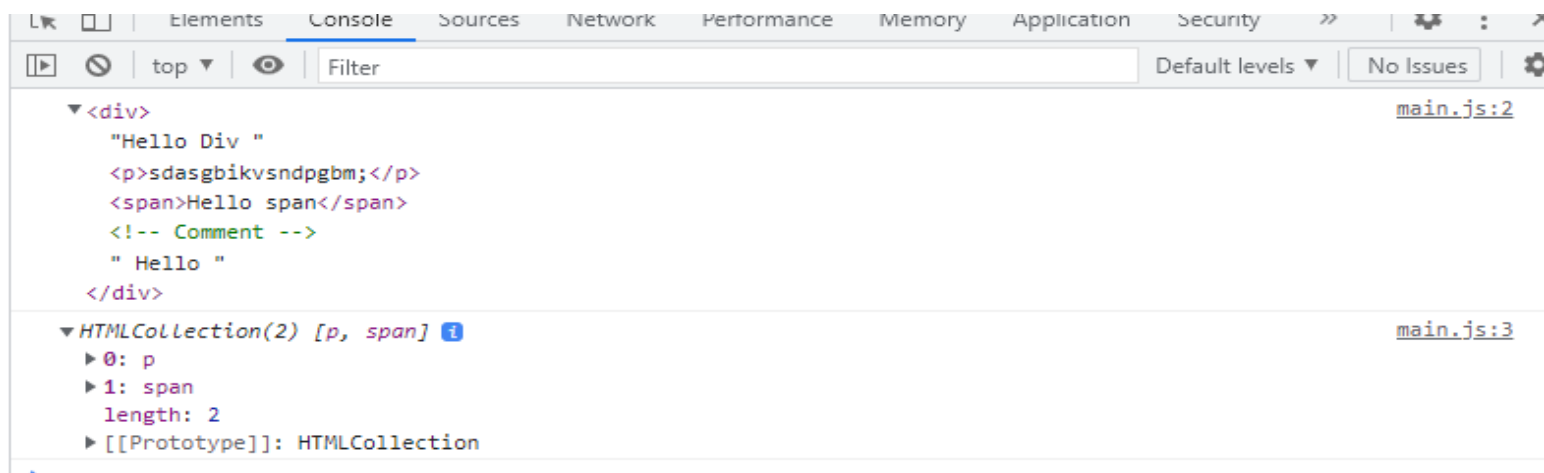

شرح :- children , childNode , FirstChild, LastChild, FirstElementchild, LastElementChild

1. ماهو الـ :-children

```
<!--Html code-->
<body>
<div>Hello Div
<p>Hello p</p>
<span>Hello span</span>
<!-- Comment -->
    Hello
</div>
<!--javaCode--
Let myElement=document.querySelector("div")
console.log(myElement)

console.log(myElement.children)
//Div يقوم بارجاع العناصر الموجودة داخل العنصر الاب وهو الـ Div
myElement.children[0].textContent="sdasgbikvsndpgbm"
//يمكنك ايضا الوصول الى عناصر الابن الاول او الثاني واجراء عمليات عليه
```

لاحظ في المخرجات ان الـ children قد قام بارجاع العناصر فقط دون محتوياتها مثل النصوص الموجوده داخلها في العنصر الاب Div وهما الـ paragraph و الـ span .



```
/* ChildNodes :- يقوم بارجاع جميع العناصر الموجودة داخل العنصر الاب بما فيهم النصوص والمسافات */
console.log(myElement.childNodes)
```

```
▼ NodeList(7) [text, p, text, span, text, comment, text] ⓘ
  ► 0: text
  ► 1: p
  ► 2: text
  ► 3: span
  ► 4: text
  ► 5: comment
  ► 6: text
  length: 7
  ► [[Prototype]]: NodeList
```

في `childNodes` يتم حساب المسافات على انها `text` عند ظهورها في المخرجات.

FirstChild: يقوم بارجاع اول عنصر ولا يهتم بنوع العنصر سواء كان `div` او `text` او مسافة او غيره وهذا يعنى انها تعمل نفس عمل `childNodes[0]`

```
<body>
  <div>Hello Div
    <p>Hello p</p>
    <span>Hello span</span>
    <!-- Comment -->
    Hello
  </div>
/*Java Code*/
console.log(myElement.firstChild)
"Hello Div "
```

lastChild: يقوم بارجاع اخر عنصر سواء كان نص او مسافة او عنصر او غيره.

```
<body>
  <div>Hello Div
    <p>Hello p</p>
    <span>Hello span</span>
    <!-- Comment -->
    Hello
  </div>
/*Java Code*/
console.log(myElement.lastChild)
```

```
" Hello "
```

>

firstElementChild: تهتم بنوع العنصر حيث ستقوم بارجاع اول عنصر موجود ولا ترجع النصوص او المسافات وانما ترجع فقط العناصر مثل `div` و `span` وغيره.

```
console.log(myElement.firstElementChild)
```

```
<p>Hello p</p>
```

>

lastElementChild:- تهتم بنوع العنصر حيث ستقوم بارجاع اخر عنصر موجود ولا ترجع النصوص او المسافات وانما ترجع فقط العناصر مثل `div` و `span` وغيره.

```
console.log(myElement.lastElementChild)
```

```
<span>Hello span</span>
```

اذا الـ `firstElementChild` , `lastElementChild` , `children` تهتم بنوع العنصر التي تقوم بارجاعه حيث انها لا ترجع المسافات ولا النصوص وانما ترجع العنصر فقط.

اما الـ `lastChild` , `firstChild` , `childNodes` لاتهتم بنوع العنصر وتقوم بارجاع اى عنصر كان سواء كان العنصر هذا نص او مسافة او عنصر حقيقى كالـ `div` , `span` وغيره.

الاحداث Events

يمكن ان يكون الحدث صفة فى العنصر بحيث انه يمكنك كتابة الامر الذى تريد تنفيذه كصفة فى العنصر فى كود الـ `html` كما فى المثال التالى :-

--> يمكن استخدام الحدث كصفة عند الضغط على الزرار التالى تنفذ لك امر معين -->!

```
<button id="btn"
onclick="console.log('clicked')">Button</button>

<hr/>

<form action="">

<input type="text"/>

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

</form>
```



top ▼



Filter

17

clicked

>

يمكن تنفيذ نفس الحدث عن طريق كتابته في كود الجافا بعد استدعائه بالـ **id** الخاص به كالتالى:-

```
let button = document.getElementById("btn")
button.onclick = function(){
    console.log("clicked")
}
```

Oncontextmenu:- تقوم باظهار قائمة وتنفيذ الامر الذى تريده عن النقر بزر
الماوس كلك يمين .

```
let myButton = document.getElementById("btn");
myButton.oncontextmenu = function() {
    console.log("clicked");
};
```

Onmouseenter:- ينفذ لك الامر الذى تريده بمجرد الوقوف بالماوس على الزر
دون النقر عليه.

```
let myButton = document.getElementById("btn");
myButton.onmouseenter = function() {
    console.log("clicked");
};
```

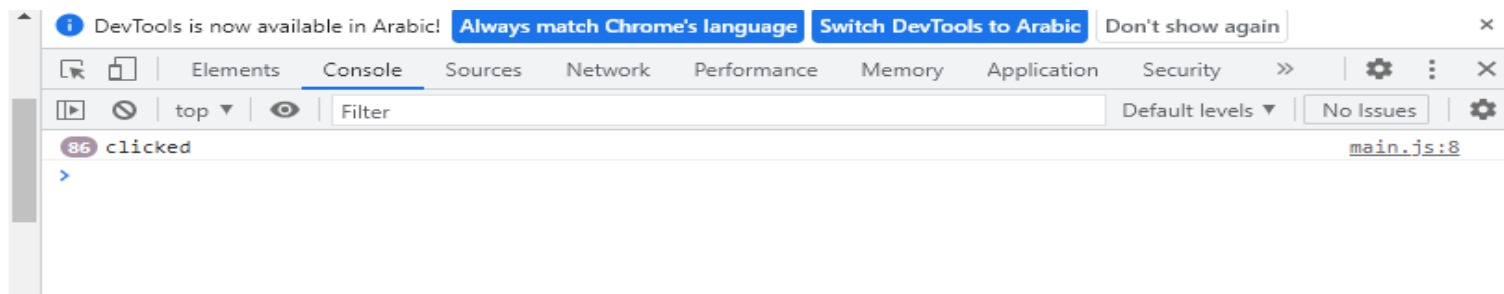
Onmouseleave:- ينفذ لك الامر الذى تريده بمجرد مغادرة الوقوف بالماوس على
الزر (يعنى اقف بالماوس على الزرار وشيله عشان الامر يتنفذ)

Onscroll:- تنفذ لك الامر الذى تريده بمجرد عمل سكرول للصفحة , حيث انه فى
المثال التالى تم تحديد طول الصفحة على انه 5000 الاف بيكسل لامكانية ظهور
السكرول فى الصفحة ثم تم كتابة الكود التالى .

```
<script> </script>
<!-- تحديد طول الصفحة -->
<style>
    body{
        height: 5000px;
    }
```

```
</style>
</head>
```

```
window.onscroll = function() {
    console.log("clicked");// print scroll in consol in case you
    scroll the window up or down
};
```



Onresize:- عندما تكبر شاشة الكونصول مثلا او تصغرها يطبع لك ام معين.

```
window.onresize = function() {
    console.log("clicked");
};
```

Onfocus:- بمجرد النقر بالماوس داخل **inputText** معين للبدء فى الكتابة يمكن كتابة امر هذا الامر يظهر رسالة ان النص يجب ان يحتوى على حروف وارقام وغيره.

onfocus

Submit Data

Onblur:- بمجرد الخروج من حقل الكتابة **inputtext** يمكن كتابة امر يظهر لك رسالة تحذيرية انك مثلا لم النص بالصيغة المطلوبة.

Onsubmit:- بمجرد النقر على ارسال النص المكتوب يمكنك بهذا الامر عمل فحص على النصوص المكتوبة مثلا قبل ارسالها واذا كانت سليمة قم بارسالها .

Validate

هو عبارة عن الشروط تقوم بوضعها عند كتابة شيء ما داخل الحقل مثل اذا كان الحقل يطلب ان البيانات المدخلة داخله تكون من نوع ايميل او ارقام او رموز وانت خالفت شروط الكتابة داخل الحقل ستظهر لك رسالة خطأ تخبرك ان البيانات يجب ان تكون من النوع المحدد .

```
// بمجرد الوقوف بالماوس على الرابط جوجل نوع الكود المستخدم
document.links[0].onmouseenter = function(event){
console.log(event)
}
```

لتطبيق ذلك على الـ form لرؤية عمل الـ `.validates`.

فى الحقل الاول :- سنقول مثلا ان لدينا متغيرين كل واحد منهم قيمته `false` ونربط قيمة المتغير بقيمة الحقل الذى سيرسل البيانات بوضع شرط يقول اذا كان قيمة الحقل ليست فارغة وطولها اقل من اويساوى الـ 10 فان قيمة المتغير ستكون `true`.

فى الحقل الثانى :- سنضع شرط اذا كان الحقل ليس فارغ فان المتغير ستكون قيمته `true`

الشرط الذى سينفذ عليه الـ `validate` سيقول اذا كان المتغير قيمته `false` لاترسل البيانات واذا كان قيمته `true` قم بارسال البيانات.

Html Code

```
<form action="">
  <input type="text" name="username" placeholder="Max 10
chars only" />
  <input type="text" name="age" placeholder="Cant Be Empty"
/>
  <input type="submit" value="Submit Data" />
</form>
<a href="https://www.google.com">Google</a>
```

JavaScript Code

```
let userInput = document.querySelector("[name='username']")
let ageInput = document.querySelector("[name='age']")
```

```
document.forms[0].onsubmit = function(ele){
    /*الوضع الطبيعي للvalid هو false لانها فارغة لم يكتب فيها شيء*/
    وعند التأكد من صحة البيانات يتم تغييرها من false الى true
    */
    let userValid = false ;
    let ageValid = false ;

    if(userInput.value !== "" && userInput.value.length <=10){
        userValid = true ;
    }
    if(ageInput.value !== ""){
        ageValid = true;
    }
    //سيتم وضع شرط بحيث اذا كان الـ valid يساوي false فلن يتم ارسال البيانات
    if (userValid===false || ageValid===false){
        ele.preventDefault();// قم بوقف عمل الكود او الحدث
    }
};
```

من المتعارف عليه انه بمجرد النقر داخل الحقل بالماوس يتم عمل مايسمى بالـ **focus** ولكن يمكن عمل ذلك بمجرد تحميل الصفحة عن طريق مايلي :-

كود الـ html

```
<form action="">
    <input class="one" type="text" />
    <input class="tow" type="text" />
    <input class="submit" value="Submit Data" />

</form>
```

كود الـ java

```
let tow =document.querySelector(".tow");
window.onload= function(){
    tow.focus();
}
```

[Google](#)

لنجرّب الآن بمجرد الخروج من الحقل يتم النقر على الرابط الموجود.

```
let one = document.querySelector(".one");
one.onblur = function(){
    document.links[0].click();
};
```

تعامل الـ `classList` مع الـ `dom`

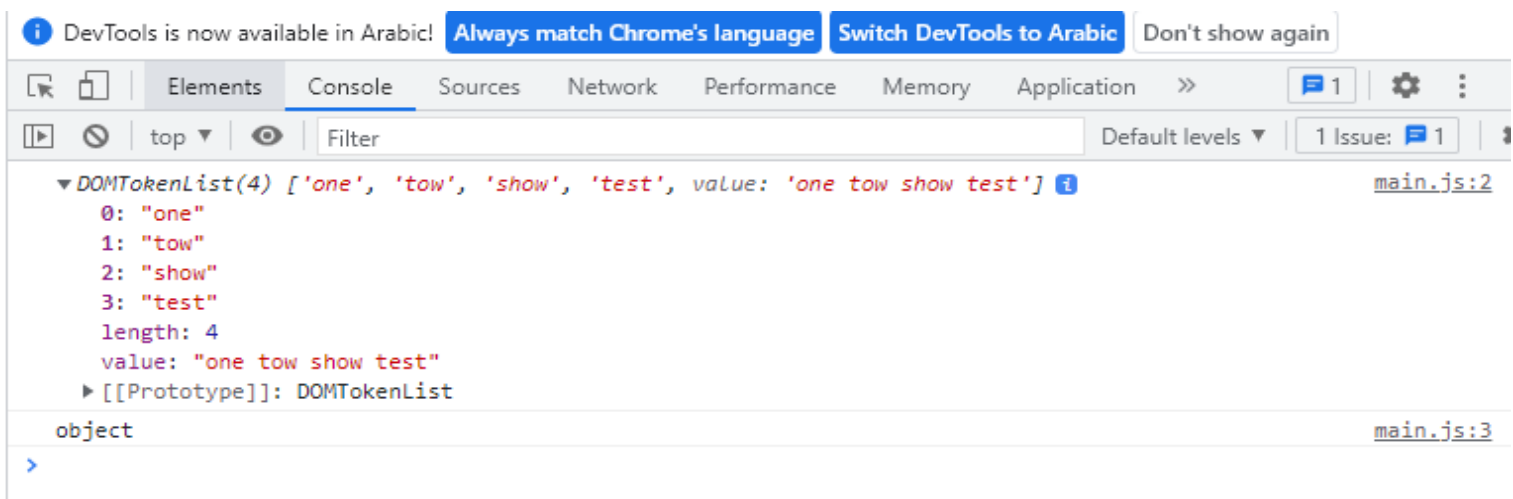
Html Code

```
<div id="my-div" class="one tow show test">Div with Many
Classes</div>
```

Java Code

```
let element = document.getElementById("my-div")
console.log(element.classList)
console.log(typeof element.classList)
```

`classList`:- هي عبارة عن قائمة تعرض لك جميع الكلاسات بالعنصر كما هو موضح في المخرجات:-



ماهو الـ `contains`:- هل العنصر يحتوي على الصفة كذا ام لا ,,,,,, مثال :-

```
console.log(element.classList.contains("hossam"))
console.log(element.classList.contains("show"))
```

false

true

>

نتطرق الان الى الـ `item`:- يطلب منك الـ `item` الائنكس الخاص بالـ `class` حيث يوضح الكود التالي اسم الكلاس للعنصر رقم 1 و 0

```
console.log(element.classList.item("0"))
console.log(element.classList.item("1"))
```

```
one
tow
```

`add`:- تستخدم لاضافة صفة او اكثر للعنصر كما هو موضح فى المثال التالى :-

htmlCode

```
<div id="my-div" class="one tow show test">Div with Many
Classes</div>
```

Java Code

```
let element = document.getElementById("my-div")
element.onclick= function (){
    element.classList.add("add-one" , "add-tow")
}
```

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
```

```
...<body> == $0
```

```
  <div id="my-div" class="one tow show test add-one add-tow">Div with Many Classes</div>
```

```
  <!-- يمكنك كتابة كود الجافا داخل السكريبت او استدعائه من ملف اخر عن طريق الكود التالى -->
```

```
  <script src="main.js"></script>
```

```
</body>
```

```
</html>
```

`Remove`:- يقوم بحذف الصفات للعنصر وهى عكس الـ `add`

```
let element = document.getElementById("my-div")
element.onclick= function (){
    element.classList.remove("tow" , "show")
}
```

Div with Many Classes

DevTools is now available in Arabic! Always match Chrome's language Switch DevTools to Arabic Don't show again

```
Elements Console Sources Network Performance Memory Application
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body> == $0
    <div id="my-div" class="one test">Div with Many Classes</div>
```

Toggle:- يقوم بحذف الصفة اذا كانت موجودة واضافتها اذا كانت غير موجودة

```
let element = document.getElementById("my-div")
element.onclick= function (){
    element.classList.toggle("hossam");
};
```

<div id="my-div" class="one tow show test hossam">Div with Many Classes</div>

cssStyling

التحكم فى خصائص العنصر من حيث اللون والحجم وغيره عن طريق الجافا.

- تم انشاء ملف main.css وربطه بصفحة الـ html عن طريق الكود التالى:-

```
<link rel="stylesheet" href="main.css" />
```

ثم تم انشاء الـ Div

```
<div id="my-div" >Div with Many Classes</div>
```

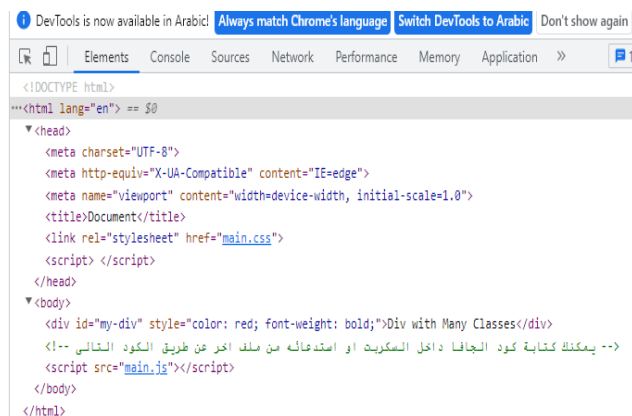
ثم تم وضع خصائص له فى صفحة الـ main.css

```
div {
    font-size: 30px;
    line-height: 2;
}
```

- كود الجافا لتغيير خصائص العنصر .

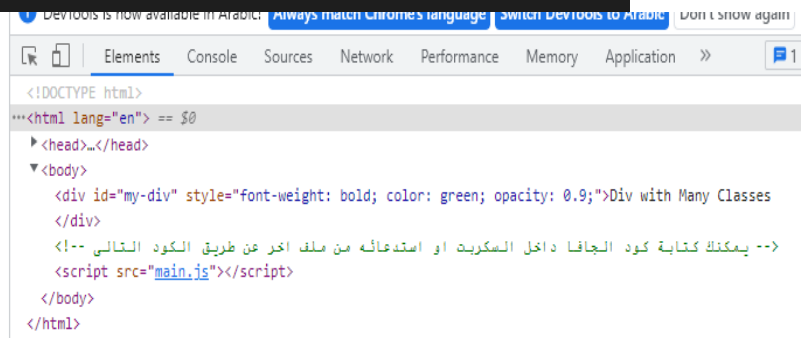
```
- let element =document.getElementById("my-div")
-
- element.style.color ="red"// لون العنصر
- element.style.fontWeight="bold" // سمك الخط
```

Div with Many Classes



```
let element=document.getElementById("my-div")
/*وضع صفات العنصر بطريقة الـ CSS العادية*/
element.style.cssText="font-weight: bold ; color: green; opacity
: 0.9";
```

Div with Many Classes



تعيين صفة CSS او ازلتها عن طريق `removeProperty` , `setProperty` -:

```
let element=document.getElementById("my-div")

element.style.removeProperty("color");// تستخدم لازالة خاصية
element.style.setProperty("font-size", "100px", "important");// اضافة
خاصية والقيمة تكون بعد الفاصله
```



Div with Many Classes

الوصول الى خصائص الـ CSS الموجودة في ملف `main.css` والتعديل عليها بطريقة الـ `removeProperty` و `setProperty` .

سيتم تحديد لينك الرابط `styleSheets[0]` الموجود في ملف الـ `html` مع تحديد رقمه وهو الصفر لانه لا يوجد غيره ثم بعد ذلك يتم تحديد `rules` وهي الخصائص الموجودة في ملف الـ `main.css` ولا يوجد الا خاصية واحدة لذا ستكون `rules[0]` ثم كتابة `style` ثم استخدام الخاصية `setProperty` او `removeProperty` . مثال -:

-:Stylesheets

```
<link rel="stylesheet" href="main.css" />
```

-:Rules

```
div {  
  font-size: 30px;  
  line-height: 2;  
}
```

الكود فى الجافا :-

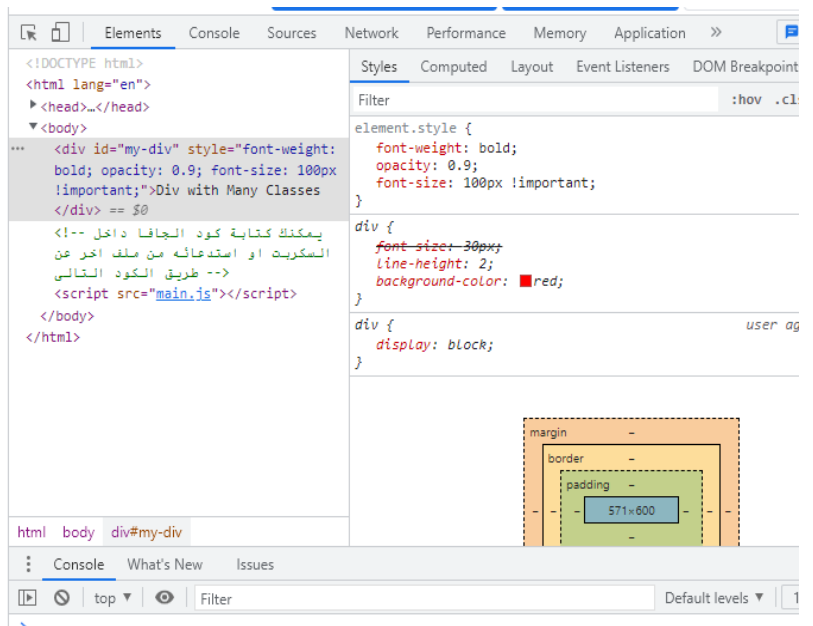
```
document.styleSheets[0].rules[0].style.removeProperty("line-height")
```

```
element.style {  
  font-weight: bold;  
  opacity: 0.9;  
  font-size: 100px !important;  
}  
  
div {  
  font-size: 30px;  
}  
  
div {  
  display: block;  
}
```

استخدام نفس الطريقة لاضافة خاصية للعنصر بالـ setProperty

```
document.styleSheets[0].rules[0].style.setProperty("background-color" , "red")
```

Div with
Many
Classes



استخدام بعض الطرق للتعامل مع العناصر

Before , after , Prepend , Append , Remove

تستخدم before لإنشاء صفة أو عنصر قبل العنصر المحدد كالمثال التالي :-

```
let element =document.getElementById("my-div")  
  
let createP = document.createElement("p")  
element.after(createP) //Div بعد Paragraph وضع
```

```
<div id="my-div">Div with Many Classes</div>  
<p></p>
```

يمكن استخدام نفس الطريقة لوضع العنصر paragraph قبل الـ Div ولكن باستخدام الـ Before بدلا من الـ after

Append:- تستخدم لوضع عنصر أو صفة داخل عنصر آخر ولكن تكون آخر شيء داخل العنصر ويمكن ان يكون الشيء الذي يتم وضعه هو صفة أو عنصر كما في المثال التالي:-

```
let element =document.getElementById("my-div")  
let createP = document.createElement("p")  
element.append(createP)  
element.append("Hello Hossam")
```

```
<div id="my-div">  
  "Div with Many Classes"  
  <p></p>  
  "Hello Hossam"  
</div>
```

Prepend:- هي عكس Append حيث تقوم بوضع المطلوب في اول العنصر وليس في اخره .

```
let element =document.getElementById("my-div")  
let createP = document.createElement("p")  
element.prepend(createP)  
element.prepend("Hello Hossam")
```

```

<div id="my-div">
  "Hello Hossam"
  <p></p>
  "Div with Many Classes"
</div>

```

تقوم الـ remove بحذف العنصر تماما من الصفحة ولكن `display:none` يمكن للمتصفح الغائها من الـ `inspect` .

```

let element = document.getElementById("my-div")
let createP = document.createElement("p")
element.remove()

```

الان نتطرق الى الانتقال بين العناصر في Dom:-

`nextSibling`:- الانتقال الى ما بعد العنصر مباشرة دون تحديد هل ماسيتم الانتقال اليه هو عنصر ام فراغ ام نص ام غيره...

htmlCode

```

<div id="my-div" >
  <span class="one">One</span>
  <!--Comment-->
  <span class="two">Two</span>
  <!--Comment-->
  <span class="three">Three</span>
</div>

```

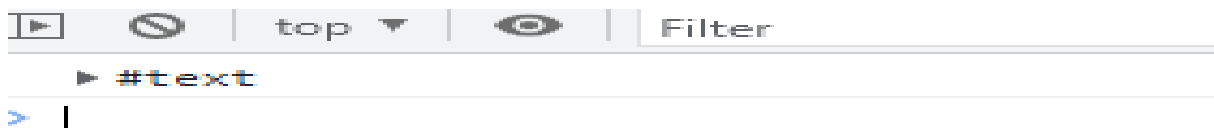
JavaCode

```

let span = document.querySelector(".two")
let span = document.querySelector(".two");
console.log(span.nextSibling);

```

لاحظ في المخرجات ان ما بعد العنصر هو نص من نوع `comment`



`nextElementSibling`:- تنتقل الى العنصر الذي يلي العنصر المحدد دون مراعاة فراغات او نصوص او غيره..

```
<span class="three">Three</span>
```

الفائدة من الـ `nextElementSibling`:- يمكن اجراء العمليات بها مثل حذف العنصر التالي :-

```
let span = document.querySelector(".tow")
span.nextElementSibling.remove()
```

لاحظ في المخرجات انه قد تم حذف الـ `span` الثالث.

```
<div id="my-div">
  <span class="one">One</span>
  <!--Comment-->
  <span class="tow">Tow</span>
  <!--Comment-->
</div>
```

الان نأتى الى الـ `previousElementSibling`:- وهى عكس الـ `nextElementSibling` حيث تقوم بالانتقال الى العنصر الذى يسبق العنصر الحالى واجراء عمليات عليه كما تريد وعدم مراعاة النصوص والفراغات وغيره وتهتم بالعناصر فقط,,, اما الذى يهتم بالنصوص والفراغات واي شىء كان قبل العنصر هو الـ `previousSibling`.

```
let span = document.querySelector(".tow")
span.previousElementSibling.remove()
```

لاحظ في المخرجات انه قد تم حذف الـ `span` الاول

Tow Three

```
Elements Console Sources Netwo
<!DOCTYPE html>
<html lang="en" == $0
  <head>...</head>
  <body>
    <div id="my-div">
      <!--Comment-->
      <span class="tow">Tow</span>
      <!--Comment-->
      <span class="three">Three</span>
    </div>
```

```
let span = document.querySelector(".tow")
console.log(span.previousElementSibling)
console.log(span.previousSibling)
```

لاحظ في المخرجات تم طباعة العنصر فى الكونصول ثم تم طباعة الـ `comment`

```
<span class="one">One</span>
```

```
#text
```

parentElement:- الوصول الى العنصر الاب لتنفيذ ماتريد.

```
let span = document.querySelector(".tow");

span.onclick = function(){
    span.parentElement.remove();
}
```

عند تنفيذ الكود السابق سيتم حذف العنصر الاب للـ Span وهو الـ Div.

DomCloning

كما درسنا في السابق انه يمكن وضع عنصر داخل عنصر اخر بطريقة الـ **appendChild** كما هو موضح في الكود التالي :-

HtmlCode

```
<p id="my-p" class="my-p">This is <span>P</span></p>
<div>Div</div>
```

JavaCode

```
let myP = document.querySelector("p");
let myDiv = document.querySelector("div");

myDiv.appendChild(myP)
```


ولكن اذا اردنا اخذ نسخة فقط من العنصر لوضعه داخل عنصر اخر دون نقل العنصر كامل وهنا يأتي دور الـ **CloneNode** التي تنسخ العنصر الذي تريد نسخه داخل عنصر اخر ولكن بدون المواصفات .

```
let myP = document.querySelector("p").cloneNode();
let myDiv = document.querySelector("div");

myDiv.appendChild(myP)
```

لاحظ في المخرجات انه قد تم نقل paragraph داخل الـ Div دون نقل الـ span معه .

```
<body>
  <p id="my-p" class="my-p">
    "This is "
    <span>P</span>
  </p>
  <div>
    "Div"
    <p id="my-p" class="my-p"></p>
  </div>
</body>
```



ونقل العنصر بجميع مواصفاته تضع فقط داخل اقوس الـ `CloneNode` كلمة `true`

```
let myP = document.querySelector("p").cloneNode(true);
let myDiv = document.querySelector("div");
myDiv.appendChild(myP)
```

لاحظ في المخرجات انه قد تم اضافة عناصر الـ `paragraph` حيث قد تم نقل الـ `span` داخل الـ `pargraf` داخل الـ `Div`

- قد يحدث مشكلة عند نسخ العناصر حيث ان الصفات ستكون متشابهة مثل الـ `id` والـ `class` وغيره وهذا قد يسبب مشاكل ولتغيير خاصية من خصائص العنصر المنسوخ :-

```
- let myP = document.querySelector("p").cloneNode(true);
- // جعلت cloneNode هذا العنصر هو العنصر المنسوخ وليس الاصلى
- let myDiv = document.querySelector("div");
- myP.id = `${myP.id}-clone` // تغيير الـ Id للعنصر المنسوخ
- myDiv.appendChild(myP)
```

- لاحظ في المخرجات انه قد تم تغيير `Id` للعنصر المنسوخ :-

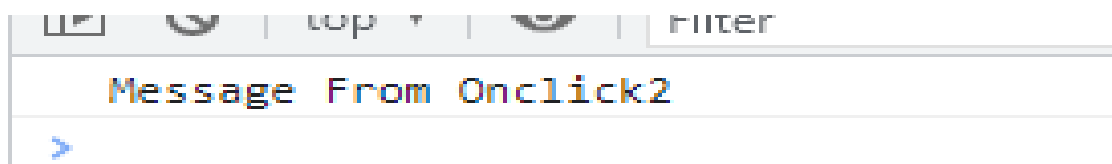
```
<body>
  <p id="my-p" class="my-p">...</p>
  <div>
    "Div"
    <p id="my-p-clone" class="my-p">
      "This is "
      <span>P</span>
    </p>
  </div>
```

addEventListener

يمكن ان يكون هناك اكثر من حدث تختار من بينهم الحدث المناسب ,,,, مثال:-

```
let myP = document.querySelector("p");
myP.onclick = tow
function one () {
  console.log("Message From Onclick1")
}
function tow () {
  console.log("Message From Onclick2")
}
```

في السطور البرمجية سيتم تنفيذ الحدث الثاني عند النقر على الـ `pargraf`.



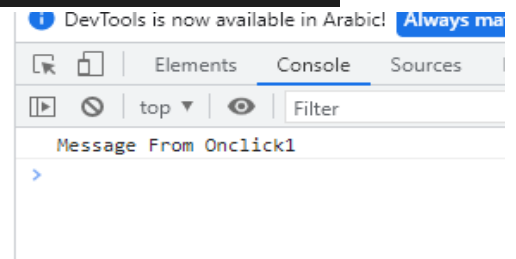
إذا تم كتابة حدثين على عنصر واحد فإن الحدث الأخير سيكون له الأولوية في التنفيذ كالتالى:-

```
let myP = document.querySelector("p");
myP.onclick = tow ;
myP.onclick = one ; // له الأولوية في التنفيذ

function one () {
    console.log("Message From Onclick1")
}
function tow () {
    console.log("Message From Onclick2")
}
```

This is P

Div



الان نأتى الى طريقة الـ `addEventListener` الذى يقوم بتنفيذ مجموعة من الاحداث بنقرة زر واحدة ولا يقتصر على حدث واحد كالمثال السابق.

فى المثال التالى يتم كتابة الحدث `click` داخل الاقواس ثم الدالة التى ستكون نتيجة النقر على الزر .

```
let myP = document.querySelector("p");
myP.onclick = tow ;

function one () {
    console.log("Message From Onclick1")
}
function tow () {
    console.log("Message From Onclick2")
}

myP.addEventListener("click" , function () { // ثم الدالة التى ستكون
    // يتم كتابة الحدث click نتيجة
    console.log("Message From Event 1")
})
```

لاحظ في المخرجات انه قد تم طباعة الحدثين بنقرة زر واحدة دون الاختصار على حدث واحد من الحدثين.

```
Message From Onclick2
Message From Event 1
```

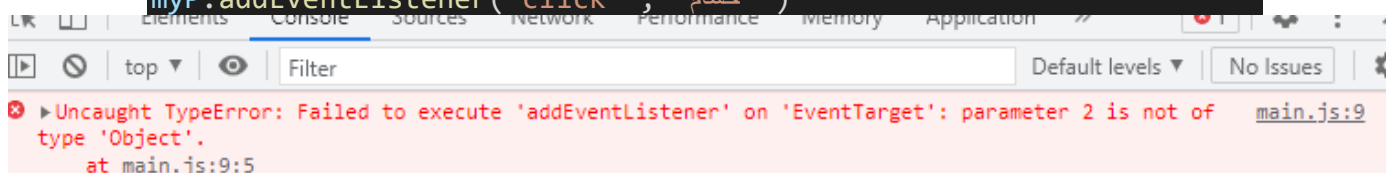
>

يمكنك ايضا وضع الدالة one او tow بدلا من الدالة الكاملة المكتوبة داخل الاقواس كالمثال التالي :-

```
let myP = document.querySelector("p");
myP.onclick = tow ;
function one () {
    console.log("Message From Onclick1")
}
function tow () {
    console.log("Message From Onclick2")
}
myP.addEventListener("click" , one ) // function one has been
added instead of full function .
```

احذر الوقوع في الخطأ التالي بحيث انك يمكن ان تفكر في كتابة اسم معين بمجرد النقر على العنصر دون وضع الاسم الذي تريد كتابته داخل دالة كالمثال التالي :-

```
let myP = document.querySelector("p");
myP.addEventListener("click" , "حسام" )
```



>

مثال على تطبيق event في عنصر ليس موجود في الصفحة .

اولا يجب ان نعرف انه من الاخطاء التي يمكن ان نقابلها هو ان يتم تنفيذ على عنصر تم نسخه نتيجة اكواد برمجية ولكنه في الاساس ليس موجود في الصفحة كالمثال التالي :-

```
let myP = document.querySelector("p");

myP.onclick = function () {
    let newP = myP.cloneNode(true)
    newP.className = "clone"
```

```

document.body.appendChild(newP)
}
// تنفيذ كود على عنصر ليس موجود في الصفحة
let cloned = document.querySelector(".clone")
cloned.onclick = function () {
  console.log("iam Cloned")
}

```

Uncaught TypeError: Cannot set properties of null (setting 'onclick')
at main.js:10:16

ولكن بطريقة الـ `addEventListener` يمكن تنفيذ الكود على عنصر ليس موجود في الصفحة
كالمثال التالي الذي يشرح عمل دالة تحتوى على حدث يقول اذا كان الغرض من الحدث هو النقر
على عنصر يحتوى على `className` باسم `clone` فقم بطباعة `iam Cloned` :-

```

document.addEventListener("click" , function(e){
  if(e.target.className==='clone'){
    console.log("iam Cloned")
  }
})

```

Clone Me

Clone Me

