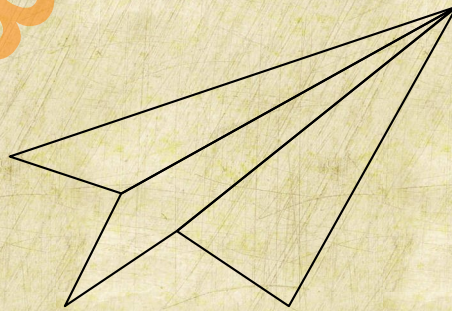


# الكافى

فى

# JavaScript

## الجزء الاول



# أبو حبيب الحسينى

# ابو حبيب الحسيني

## ملحوظة مهمة جدا

ان وجود الكلمات الانجليزية فى وسط الجمل العربية ينقل بعض الكلمات من مكانها فتظهر الجمل بشكل غير صحيح ويصعب فهما وهذا عيب فى الترميز (يو تى اف)

مثال على هذا الكلام

• أو في كليهما HTML صفحة <head> أو <body> يمكن وضع الكود في

لاحظ هنا ان الجملة اصبحت غير مفهومة الى حد ما او غير مرتبة بشكل صحيح لان بعض الكلمات نقلت من مكانها بسبب وضع كلمات انجليزية وسط الجمل العربية فافى مثل هذه الحالات حاول ان تستنتج الجمله بنفسك وتفهما

حاولنا تقليل هذا العيب قدر المستطاع باستخدام بكتابة المصطلحات الانجليزية باللغة العربية مثل (سي اس اس) او (نود جى اس) وكذلك نقلنا اتجاة الصفحة من اليسارى الى اليمين لتفادى هذا العيب وللأسف لم تتم المعالجة بنسبة مئة بالمئة

# ابو حبيب



## فهرس الكتاب

3	ملحوظة مهمة جدا
5	فهرس الكتاب
15	مقدمة قصيرة عن جافاسكريبت
17	ماذا ستعلم فى هذه السلسلة
18	ما هى برامج جافاسكريبت المستقلة
19	مميزات جافاسكريبت فى برمجة التطبيقات المستقلة
19	والسيرفرات والالعب
19	الميزة الاولى
20	الميزة الثانية
21	الميزة الثالثة
22	الميزة الرابعة
23	الميزة الخامسة
24	الميزة السادسة
25	الميزة السابعة
26	الميزة الثامنة
27	عيوب جافاسكريبت
28	ملحوظة
28	فوائد التعلم من خلال الأمثلة
28	يعنى باختصار
29	كيف استخدام جافاسكريبت فى تغير المحتوى
30	تغيير الخصائص
30	كيف تغيير قيم الخصائص
31	كيف اخفاء العناصر
31	كيف اظهار العناصر المخفية
32	<script> العلامة
32	وظائف وأحداث
32	كيف اضافة بلوكات من الاكواد فى نفس الصفحة
33	قسم الهد او راس الصفحة
33	اكتب جافاسكريبت داخل البودى واستدعيها من داخل البودى
34	كتابة جافاسكريبت فى ملف خارجى
35	مزايا جافاسكريبت فى ملف خارجى
35	المراجع الخارجية
36	دوال الاخراج
36	دوال تمكّنك من عرض البيانات
36	HTML استخدام المحتوى الداخلى
37	كيف كتابة البيانات فى الصفحة
38	كيف اظهار رسالة للمستخدم
39	كيف التسجيل داخل الكونسل
40	الطباعة الخارجية فى جافاسكريبت

تعريف البيانات في جافاسكريبت.....	40
كيف تكوين البيانات البرمجية.....	41
كيف الاعلان عن متغيرات عديدة.....	41
المسافات الفارغة.....	42
طول السطر في جافاسكريبت وفواصل الأسطر.....	42
كتل الاكود.....	43
الكلمات المحجوزة في جافاسكريبت.....	43
طرق انشاء المتغيرات.....	44
التعامل مع القيم.....	44
التعامل مع الحروف.....	45
المتغيرات.....	45
المعاملات الحسابية.....	45
التعبيرات.....	46
الكلمات المحجوزة.....	46
التعليقات.....	47
المعرفات او الأسماء.....	47
ملحوظة.....	48
احظر حساسة حالة الأحرف.....	48
تكوين الجمل.....	48
ترميز الكتابة.....	49
انشاء التعليقات.....	49
كيف انشاء تعليقات سطر واحد.....	49
كيف انشاء تعليقات متعددة الأسطر.....	50
استخدام التعليقات لمنع التنفيذ.....	50
متغيرات جافاسكريبت.....	51
ما هي المتغيرات؟.....	52
متى استخدام الكلمة var.....	53
متى استخدام الكلمة const.....	53
استخدام معملات الجبر.....	53
ملحوظة.....	54
ما هي معرفات جافاسكريبت.....	54
ملحوظة.....	54
عامل التعيين.....	54
ملحوظة.....	55
أنواع البيانات السائدة.....	55
كيف الإعلان عن متغير.....	55
ملحوظة.....	56
كيف عمل اعلان واحد، لجميع المتغيرات.....	56
كيف جعل القيمة غير محدد.....	57
إعادة الإعلان عن متغيرات.....	57
ملحوظة.....	58
انشاء العمليات الحسابية.....	58

ملحوظة.....	59
استخدام علامة الدولار \$.....	59
جافاسكريبت و الاندرسكول.....	60
احظر هذا الخطا عند استخدام الكلمة Let.....	60
ما هو نطاق الكتلة {}.....	61
إعادة الإعلان عن المتغيرات.....	62
إعادة الإعلان للمتغيرات.....	62
كيف استخدام المتغيرات قبل الاعلان عنها.....	64
استخدام الكلمة const.....	64
خصائص الكلمة كونست.....	65
يجب أن يتم تعيينه.....	65
const متى تستخدم.....	65
الكائنات والمصفوفات.....	66
تطبيق عملي على المصفوفات.....	66
مثال اخر على الكائنات.....	67
نطاق المتغير داخل الكتلة.....	67
كيف إعادة الإعلان عن المتغير.....	68
المزيد عن الرفع واعادة الاعلان.....	69
أنواع معاملات جافاسكريبت.....	71
معاملات جافاسكريبت الحسابية.....	71
معاملات تعيين جافاسكريبت.....	72
إضافة ودمج النصوص فى جافاسكريبت.....	73
إضافة نصوص وأرقام.....	74
معاملات المقارنة.....	74
عوامل المنطقية فى ووضع الشروط.....	75
معاملات الانواع.....	75
معاملات Bitwise.....	75
المعاملات الحسابية.....	76
كيف استخدام المعاملات الحسابية.....	76
مزيد من الامثلة على العمليات الحسابية.....	76
ما هى المعاملات.....	77
كيف الاستخدام فى عملية الجمع.....	77
الطرح.....	78
الضرب.....	78
القسمة.....	78
بقية القسمة.....	79
الجمع بالزيادة الثابتة.....	79
الطرح.....	79
الأس.....	80
أسبقية المشغل فى التنفيذ.....	80
ملحوظة.....	81
تعرف على المشغلات بالامثلة.....	81

مشغلات التعيين فى جافاسكربت.....	82
مشغلي مهمة التحول.....	82
Bitwise مشغلي تعيين.....	82
مشغلي التعيين المنطقي.....	82
= المشغل.....	82
=+ المشغل.....	83
=- المشغل.....	83
=* المشغل.....	84
**= المشغل.....	84
=/ المشغل.....	84
=% المشغل.....	84
=>> العامل.....	85
=<< المشغل.....	85
=>>> المشغل.....	85
=& المشغل.....	86
=  المشغل.....	86
=^ المشغل.....	86
=&& المشغل.....	87
=   المشغل.....	87
??= المشغل.....	88
أنواع البيانات العامة.....	88
أمثلة على انشاء انواع من البيانات العامة.....	89
ملحوظة.....	89
مفهوم أنواع البيانات.....	89
ملحوظة.....	90
أنواع جافاسكربت الديناميكية.....	91
نصوص جافاسكربت.....	91
الأرقام فى جافاسكربت.....	92
الأسية فى الجبر.....	92
ملحوظة.....	93
نوع البيانات BigInt.....	94
جافاسكربت و الاساليب المنطقية.....	94
المصفوفات فى جافاسكربت.....	95
الكائنات فى جافاسكربت.....	95
كيف معرفة الانواع للبيانات.....	96
الاعلان عن نوع غير معرف وليس له قيمة.....	96
القيم الفارغة.....	97
الوظائف فى جافاسكربت.....	97
كيف بناء جملة الوظيفة.....	97
كيف استدعاء الوظيفة.....	98
التحكم فى الارجاع من داخل الوظيفة.....	98
لماذا نستخدم الوظائف؟.....	99



استخدام الوظيفة لارجاع ناتج العملية.....	99
كيف استخدام الوظائف كقيم متغيرة.....	100
نطاق المتغيرات المحلية.....	100
كائنات جافاسكريبت.....	101
تعريف الكائن.....	102
خصائص الكائنات.....	102
الوصول إلى خصائص الكائن.....	102
طرق الكائن.....	103
كيف الاشارة الى الكائن بدون ذكر اسمة.....	104
تشير الكلمة المحجوزة إلى كائن this ،.....	104
ملحوظة.....	104
أنظر أيضا:.....	104
كيف استخدام كلمة ديس.....	104
كيف الوصول الى اعضاء الكائن.....	104
لا تستخدم هذه الطريقة.....	105
أحداث جافاسكريبت.....	105
HTML أحداث.....	106
الأحداث الشائعة لـ HTML.....	107
معالجات أحداث جافاسكريبت.....	107
التعامل مع النصوص.....	108
كيف معرفة عدد حروف النصوص.....	109
كيف استثناء بعض الحروف.....	109
كيف تحديد طول السطر.....	110
طريقة استخدام النصوص ككائنات.....	111
دوال النصوص.....	113
الدالة length.....	114
كيف استخراج أجزاء من النص.....	114
طرق التقطيع الدالة الاولى.....	114
ملحوظة.....	115
تابع طرق تقطيع النصوص.....	116
كيف البحث والاستبدال فى النصوص.....	117
ملحوظة1.....	117
ملحوظة2.....	118
ملحوظة3.....	119
دالة replaceAll().....	119
ملحوظة4.....	119
التحويل إلى الأحرف الكبيرة والصغيرة.....	119
دالة toUpperCase ().....	120
دالة toLowerCase().....	120
الدالة concat.....	120
ملحوظة.....	121
دالة Trim().....	121

TrimStart () الدالة.....	121
TrimEnd () دالة.....	122
دوال اخرى للاضافة.....	122
PadStart () دالة.....	122
ملحوظة.....	123
دعم المتصفحات.....	123
PadEnd () الدالة.....	123
ملحوظة.....	124
دعم المتصفحات.....	124
كيف استخراج أحرف معينة من النص.....	124
charAt () الدالة.....	124
charCodeAt() الدالة.....	125
جلب الحرف بالفهرس.....	125
ملحوظة.....	125
تحويل نص إلى مصفوفة.....	126
الطريقة العادية للتحويل.....	126
كتاب المرجع الكامل فى الجافاسكربت.....	127
كيف جلب الفهرس للحرف.....	127
ملحوظة.....	127
lastIndexOf () الدالة.....	128
استخدام دالة البحث.....	129
ملحوظة.....	129
كيف البحث الاحترافى فى النصوص.....	129
ملحوظة.....	130
matchAll () الدالة.....	130
ملحوظات.....	131
كيف معرفة نص البحث موجود ام لا.....	131
ملحوظات.....	132
طرق اخرى للتشيك على النصوص.....	132
ملحوظات.....	133
كيف البحث فى نهاية النصوص.....	133
ملحوظات.....	134
الاقتباس الحر.....	134
اقتباسات داخل الاقتباس الحر.....	135
تابع الاقتباس الحر.....	135
ملحوظة.....	135
مثال اخر لستخدام الاقتباس الحر.....	136
طريقة دمج اكواد داخل النصوص.....	136
قوالب HTML.....	137
دعم المتصفحات.....	137
كتاب المرجع الكامل فى جاسكربت.....	137
طرق التعامل مع الارقام.....	137

الاعداد الصحيحة.....	138
الاعداد العشرية.....	139
إضافة الأرقام والنصوص معا.....	139
النصوص الرقمية.....	141
الكلمة NaN.....	142
القيمة الا نهائية.....	144
السداسي عشري.....	145
ألرقام ككائنات.....	146
لا ينصح باستخدامها.....	146
BigInt على.....	147
امثلة على.....	147
مكون الاعداد الصحيحة.....	147
كيفية إنشاء نوع البيانات BigInt.....	148
تابع: BigInt.....	148
مشغل BigInt.....	149
ملحوظات.....	149
الكسور العشرية الكبيرة.....	150
استخدم الهيكس، أو الكتال أو الثنائي فى BigInt.....	150
تقريب المكون.....	150
دعم المتصفحات.....	151
الحد الأدنى والحد الأقصى للأعداد الصحيحة الآمنة.....	151
طرق او دوال الأرقام.....	151
طريقة Number.isInteger().....	151
طريقة Number.isSafeInteger().....	152
طرق ارقام جافاسكريبت.....	152
طريقة toString().....	152
طريقة toExponential().....	153
طريقة toFixed().....	153
طريقة toPrecision().....	154
طرق اخرى لجلب القيمة.....	154
لتحويل المتغيرات إلى أرقام.....	155
الطريقة الاولى Number().....	155
كيف تحويل التاريخ Number().....	155
الطريقة الثانية parseInt().....	156
الطريقة الثالثة parseFloat().....	156
شرح اهم دوال الارقام.....	157
لا يمكن استخدام الأساليب الرقمية فى المتغيرات.....	157
دالة Number.isInteger().....	158
دالة Number.isSafeInteger().....	158
دالة Number.parseFloat().....	158
ملحوظة.....	159
الدالة Number.parseInt().....	159
كتاب المرجع الكامل لجافاسكريبت.....	160

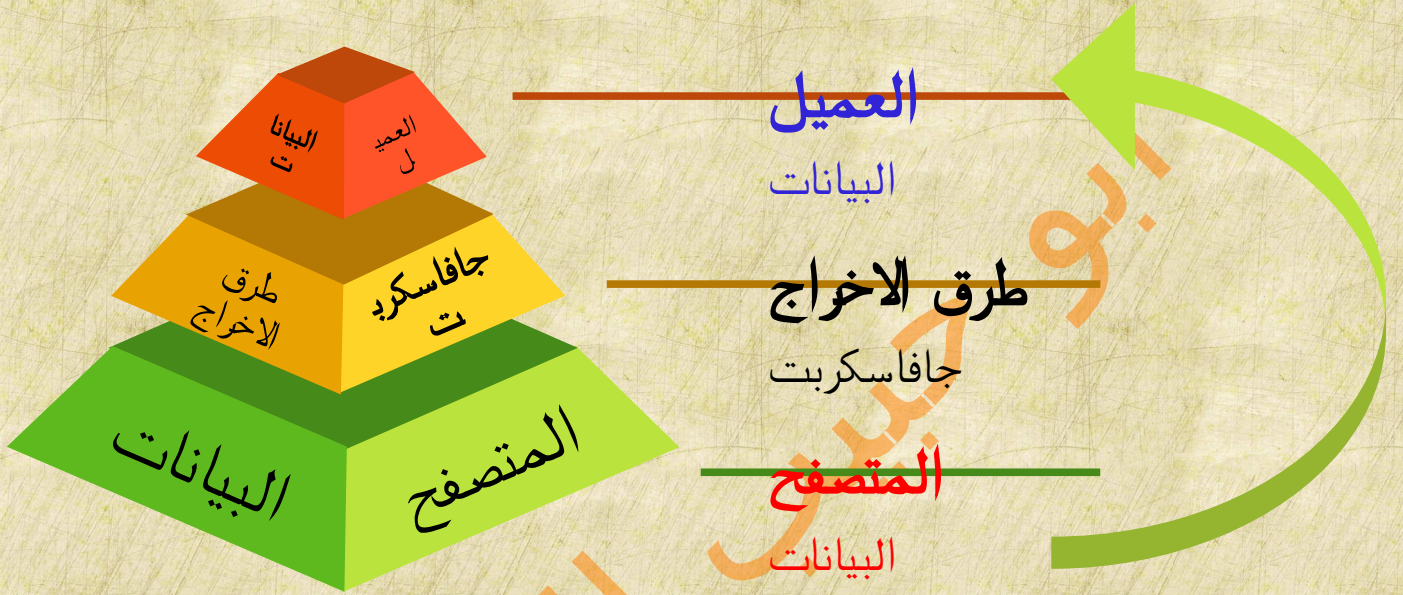
خصائص الارقام.....	160
خاصية ابسيلون.....	160
ملحوظة.....	161
خاصية MAX_VALUE.....	161
لا يمكن استخدام خصائص الأرقام في المتغيرات.....	161
خاصية MIN_VALUE.....	162
خاصية MAX_SAFE_INTEGER.....	162
خاصية MIN_SAFE_INTEGER.....	162
خاصية POSITIVE_INFINITY.....	163
خاصية NEGATIVE_INFINITY.....	163
شرح NaN.....	164
المصفوفات.....	164
لماذا استخدام المصفوفات؟.....	165
كيف إنشاء مصفوفة.....	165
استخدام الكلمة كونست.....	166
الوصول إلى عناصر المصفوفة.....	167
كيف تغيير عناصر المصفوفة.....	167
الوصول إلى المصفوفة بلكاملة.....	167
المصفوفات هي كائنات.....	168
يمكن أن تكون عناصر المصفوفة كائنات.....	168
خصائص المصفوفة وطرقها.....	169
كيف معرفة طول المصفوفة.....	169
كيف الوصول إلى العنصر الأول.....	169
كيف الوصول إلى العنصر الأخير.....	170
كيف عرض جميع العناصر.....	170
كيف إضافة عناصر في نهاية المصفوفة.....	171
المصفوفات الترابطية.....	172
الفرق بين المصفوفات والكائنات.....	173
متى تستخدم المصفوفات و متى تستخدم الكائنات.....	173
كيف استخدام new Array().....	173
كيفية التعرف على المصفوفة.....	174
طرق المصفوفات.....	175
اولا تحويل المصفوفات إلى نصوص.....	175
تعديل المصفوفات.....	176
كيف ازالة العنصر الاخير من المصفوفة.....	176
كيف اضافة عنصر جديد في النهاية.....	177
اساليب الحذف.....	177
كيف حذف العنصر الاول من المصفوفة.....	178
كيف اضافة عنصر الى البداية والغاء الاراحة للمفاتيح.....	178
كيف استبدال قيمة عنصر معين.....	179
كيف استخدام مجموع العناصر للاضافة.....	179
حذف عناصر من المصفوفة.....	180

كيف دمج المصفوفات.....	180
إضافة عناصر في وسط المصفوفة.....	181
كيف إضافة عناصر في وسط المصفوفة.....	181
إزالة عناصر من محددة في المصفوفة.....	182
كيف تجزئة المصفوفة.....	182
ملحوظة.....	183
كيف تحويل المصفوف الى نص عادى.....	184
ملحوظة.....	184
كيف إيجاد القيم القصوى والصغرى في المصفوفة.....	184
كيف الفرز والترتيب للمصفوفات.....	184
أشهر طرق الفرز للمصفوفة.....	185
كيف عكس ترتيب المصفوفة.....	185
كيف عمل ترتيب رقمى للمصفوفة.....	185
وظيفة المقارنة.....	186
فرز مصفوفة بترتيب عشوائى.....	187
طريقة فيشر بيتس.....	187
ابحث عن أعلى أو أدنى قيمة للمصفوفة.....	188
Math.max() استخدام.....	189
Math.min() استخدام.....	189
Min / Max طرق خاصة بي.....	189
فرز مصفوفات الكائنات.....	190
تكرار مصفوفة جافاسكربت.....	191
forEach ().....	191
استخدام دالة الخريطة.....	192
كيف تصفية المصفوفة والبحث فيها.....	193
كيف التقليل والتصفية للمصفوفات.....	194
إزاحة المصفوفة الى اليمين.....	195
تابع طرق المصفوفة.....	196
طريقة اخرى للبحث.....	197
كيف التحكم عن طريق الفهارس.....	198
lastIndexOf () مصفوفة جافاسكربت.....	198
طريقة اخرى للبحث.....	199
دعم المتصفح.....	200
دالة findIndex ().....	200
دعم المتصفح.....	200
Array.from().....	201
دعم المتصفح.....	201
كيف الحصول على مفاتيح المصفوفة.....	201
دعم المتصفح.....	202
إدخالات المصفوفة.....	202
دعم المتصفح.....	202
التتضمن في المصفوفة.....	203

دعم المتصفح.....	203
استخدام كوندست مع المصفوفة	203
(ES6) إيكماسكرت 2015.....	203
لا يمكن إعادة تعيينها.....	204
المصفوفات ليست ثوابت.....	204
يمكن إعادة تعيين العناصر.....	204
دعم المتصفح.....	205
تم تعيينه عندما أعلن.....	205
نطاق كتلة كوندست.....	206
إعادة تعريف المصفوفات.....	206
كتاب المرجع الكامل لجافاسكرت.....	208

أبو حبيب الحسيني

جافاسكربت هي وسيلة قوية وسريعة للتحكم في طرق الاخراج  
البيانات الى العميل والتحكم في خصائص العناصر وتغيرها في اى وقت



## مقدمة قصيرة عن جافاسكربت

بسم الله الرحمن الرحيم

جافاسكربت هي لغة البرمجة الأكثر شعبية في العالم لكونها اللغة المعترف بها من منظمة الويب العالمية كلغة أساسية لجميع متصفحات الانترنت في العالم حيث انها مدعومة من مئة بالمئة من متصفحات الانترنت الحديثة في جميع انحاء العالم وجميع انظمة التشغيل المختلفة الحديثة وجميع الشركات المنتجة للمتصفحات على مستوى العالم كله بلا استثناء وجميع انظمة تشغيل السرفرات بلا استثناء لا يوجد سيرفر ويب على وجه الارض لا يدعم جافاسكربت يعنى اى تشغيل محلى لاي متصفح سواء كان هذا للموبايل او الكمبيوتر فهي مدعومة وهي لغة مهمه جدا و يجب تعلمها مهما كان تخصصك في البرمجة لانك لاسيما ستحتاج اليها عند التعامل الاحترافى مع اى متصفح انترنت محلى او اى سيرفر مهما كانت لغته الام وهي سهله التعلم وبسيطة ومختصرة وتستطيع من خلالها انتاج تطبيقات مستقلة قائمه بذاتها لا تعتمد على اى سيرفر او متصفح ناهيك عن استخدامها في الجرافيكس داخل برامج ادوبى وغيرها تستطيع تخصيص سكربتات داخل ادوبى افتر افكت للخدع السينمائية وكذلك انشاء اسكربتات خاصه بالجافاسكربت داخل ادوبى فوتوشوب وسنشاء كتاب خاص باذن الله تعالى لاستخدام جافاسكربت من داخل الفوتوشوب وكيف تكتب اسمك على الاف الصور وتغير الخلفيات والالون لامئات الاف من الصور بضغطه زر واحدة بالجافاسكربت من داخل برنامج الفوتوشوب وكذلك جميع برامج ادوبى وكل التطبيقات التى ستخدمها فختصرا جافاسكربت تعمل فى اى مكان غير امكانتها المزملة فى التحكم فى المتصفحات للانترنت وكيف انشاء تطبيقات حرة تعمل على اى نظام تشغيل تتخيلة خاص موبايل او كمبيوتر وذلك باستخدام اطار عمل (النود جى اس) هذا الاطار الذى اعتبره انا قنبلة متفجرة فى عالم التقنيات و البرمجيات وهو بمثابة استقلاله لجافاسكربت حيث يجعلها تعمل خارج المتصفح ويتم تحويلها الى لغة البايونارى الثنائيه التى يفهما الكمبيوتر



## ماذا ستتعلم فى هذه السلسلة

ستتعلم من هذه السلسلة (سلسلة الكافة فى الجافاسكربت) كل استخدامات اللغة من البداية إلى المستوى الاحترافى باذن الله تعالى وهى خمس اجزاء قابل للزيادة منهم اربع اجزاء متتالية والجزء الخامس عبارة عن مرجع شامل لكل دوال اللغة بلا استثناء سيغطى هذا المرجع كل طرق وخصائص ودوال وكلاسات ومكتبات اللغة كاملة باذن الله تعالى

## ما هي برامج جافاسكربت المستقلة

برنامج الكمبيوتر المستقل هو عبارة عن قائمة من "التعليمات" التي سيتم "تنفيذها" تلقائياً بواسطة الكمبيوتر فقط ولا تحتاج الى وسيط يعنى برنامج قائم بذاته لا يحتاج الى تطبيقات لتشغيله والمقصود بالوسيط هنا هو المتصفح او السيرفر او برنامج مساعد مثل مشغل الفلاش الذى لا تستطيع تنفيذ برامج الفلاش بدون استخدام جافاسكربت مستقلة تعمل بذاتها على اى جهاز يجب تنصيب (النود جى اس) اولا وهو فريمورك سيجعلك تستخدم جافاسكربت خارج المتصفح يتم تنفيذها تلقائياً بدون وسيط لتنتج تطبيقات لجميع انظمة التشغيل سواء الموبايل او الكمبيوتر و سيجعلك تستخدمها ايضا كاسيرفر مثل ال (بى اتش بى) تمام لتنتج مواقع ويب تعتمد على تخزين البيانات واستدعائها كالفيسبوك واليوتيوب وغيرهم

في لغة البرمجة، تسمى تعليمات البرمجة هذه البيانات الثنائية وسميت ثنائية لانها تتكون من الزيرو والواحد فقط لن نتطرق الى هذا الان سننتج كتاب خاص لشرح لغة البينارى الثنائية ان شاء الله تعالى وهى اللغة التى يفهمها اى كمبيوتر والتى يتم تحويل جميع لغات البرمجة باختلاف انواعها الى هذه اللغة المكونة من الزيرو والواحد فقط وسنتعرف باذن الله تعالى فى كتاب لغة البينارى كيف يفهم الكمبيوتر البيانات .  
 . بالتفصيل الغير ممل

يعنى اختصارا برنامج جافاسكربت المستقل عبارة عن قائمة من البيانات البرمجة يتم تنفيذها تلقائياً بدون تطبيقات مساعدة حيث انها تترجم للغة الكمبيوتر ترجمة فورية وهذا .  
 . يسمى ال (رن تايم) ترجمة وقت التشغيل

انما يتم تنفيذ برامج جافاسكربت المدعومة فى صفحات الويب والمعرف بها من ،  
 . منظمة الويب العالمية بداخل متصفحات الويب كاجزاء من المتصفح

## مميزات جافاسكربت فى برمجة التطبيقات المستقلة والسيرفرات والالعاب

للاسف الشديد منتشر بين الناس ان لغة جافا وكتلن و سى شارب وغيرهم هما افضل لغات لبرمجة الاندريد وغيره من انظمة تشغيل الموبايل وهى معلومة غير صحيحة كما سنعرف الان

اولا ساذكر لك مميزات جافاسكربت التى تنفرد بها فى برمجة تطبيقات مستقلة وبرمجة السيرفرات والمواقع وبرمجة انظمة تشغيل الموبايل و الكمبيوتر و المميزات الغير موجودة فى اللغات الكبرى مثل جافا وروبي وبايثون وسى شارب وغيرها بمعنى انها مميزات حصرية لجافاسكربت سنتعرف عليها الان واحدة تلو الاخرى ان شاء الله تعالى

### الميزة الاولى

هى اكواد جافاسكربت الثابته ومعنى ثابتة اى انها لا تتغير على كل انظمة التشغيل الموبايل والكمبيوتر والمتصفحات والسيرفرات وكل شى وهذا يعنى انك ستتعلم مرة واحدة فقط اكواد تعمل على كل شى تقريبا مما يجعل الامور فى غاية السهولة وهذا على عكس اللغات الاخرى فكل نظام تشغيل له مكتباته الخاصة التى يجب ان تستدعيها وتدرس اكواد جديدة تعمل عليها لكل نظام يعنى مثلا اذا كنت مبرمج بايثون المصنفة ك اسهل لغة فى العالم لا يمكن ان ترمج بايثون للاندرويد الا اذا قمت باستدعا مكتبة تسمى ( اندرويد ) وهى مكتبة كبيرة جدا ومتشعبة ناهيك عن عشرات المكتبات المساعدة لها كل هذه المكتبات لا تعمل الا لنظام اندريد فقط ثم اذا اردت البرمجة بايثون لليفون له مكتباته الخاصة ايضا وعشرات المكتبات المساعدة وكذلك جميع لغات البرمجة مما يزيد الامر تعقيدا وصعوبة

## الميزة الثانية

جافاسكربت تستخدم لغة (السي اس اس) فى تنسيقات واجهة المستخدم فى التطبيقات المستقلة والسيرفات والمواقع وكل شى وهى اسهل لغة تنسيق واجهات فى العالم تكتب اسم الخاصية وامامها القيمة فقط كانك تكتب جواب لشخص لا يوجد فيها غموض مثل لغات البرمجة المعهودة اعتقد انه لا يوجد اسهل منها فى تاريخ البرمجة. فى تنسيقات الواجهات فلقد درست انا جميع لغات الدت نت من ميكروزفت حتى لغات البيسك القديمة درستها بدايتا من فيجوال بيسك 6 وحتى اطارات الدت نت واعلم جيدا معنا كلمة صعوبة تنسيق الواجهات للتطبيقات المستقلة خصوصا فى سي شارب وكل اصدرات فيشوال بيسك القديم والحديثة وجافا وجميع عائلة السي وغيرها حتى بايثون المصنفة انها اسهل لغة فى العالم تنسيق الواجهات فيها اصعب من (السي اس اس) وحتى لغات انكل قوكل الذى يدعى كذبا انها سهلة مثل لغة (ار) ليست اسهل من (السي اس اس) فى تنسيقات واجهة المستخدم فلن تجد منافس لهذه الميزة لا تبحث

## الميزة الثالثة

جافسكربت فى برمجة التطبيقات المستقلة تستخدم ادوات الكنترول الخاصة بل (اتش تى ام ال) والمقصود بادوات الكنترول يعنى (ادوات التحكم) مثل الازرار ومربعات الحوار والرسائل المنبثقة ومربعات الادخال وغيرها هذة تسمى ادوات التحكم واختصرا لن تجد فى تاريخ البرمجة القديم والحديث والماضى والحاضر والذى لم يحضر ادوات تحكم اسهل من ادوات (اتش تى ام ال) حتى ادوات (تى كى انتر) التابعه لاسهل لغة فى العالم وهى بايثون بتصنيفات عمو قوغل ليست اسهل من ادوات (اتش تى ام ال) ناهيك عن قوة هذة الادوات بعد دعم (النود جى اس) الذى لا حصر لمستودعاته البرمجية وسنتصدر له سلسلة كاملة ان شاء الله المتعالى

## الميزة الرابعة

قام موقع (جيتهاپ) بتصنيف جافاسكربت فى برمجة السيرفات باستخدام فريمورك (نود جى اس) بانها اسرع ريسبونس وريكويست فى العالم يصل الى 600 ارسال فى الثانية الواحدة وهذا اسرع من (بى اتش بى) عشر مرات والسر الكامن فى هذة السرعة الجبلة و الرهيب هى ان برمجة السيرفات فى الخوادم الاخرى تعتمد اعتماد كلى على (بيروسيسور) السيرفر يعنى بختصار اذا اعداد كبيرة جدا دخلت بالملايين الى الموقع سيكون حمل كبير جدا على السيرفر وسيختفى الموقع من على الانترنت بمجرد سقوط السيرفر من كثرة التحميل عالية ولكن مع سيرفر جافاسكربت الامر مختلف تماما فتستطيع تخصيص جزء كبير جدا من كود جافاسكربت ليطبق على اجهزة العملاء مستخدما ملايين البريسوسيرات للعملاء الذين دخلو الى الموقع بسبب دعم جميع متصفحات الويب بلا استثناء لأكواد جافاسكربت مما يؤدى الى سرعة كبيرة جدا فى التنفيذ ايا كان نظام تشغيل العميل لا تشغل بالك بهذا الامر تاكد ان متصفح العميل يدعم جافاسكربت بنسبة مئة بالمئة فكل المتصفحات الحديثة تدعمها بلا استثناء وهذا يجعل (بيروسيسور) السيرفر فى ثبات وقوة كبيرة وذلك سيرفر (نود جى اس) هو اخف سيرفر فى العالم و فشل اللغات الاخرى فى منافسة فى سرعة التنفيذ التى تكمن فى هذة الميزة المنتزعة من الخوادم الاخرى وميزات اخرى سنذكرها فيما بعد ان شاء الله تعالى حتى لغة (السى بلص بلص) المشهورة بالقوة والسيطرة و السرعة لانها تفسر للغة الالة مباشرة لم تستطع منافسة (نود جى اس) فى السرعة بسبب ميزة دعم المتصفحات المنفردة بها جافاسكربت فسرعة (نود جى اس) ليس لها منافس لتصبح جافاسكربت هى اللغة الوحيدة فى العالم التى تستطيع تسخير و توظيف بيروسوسيرات العملاء عبر المتصفحات

## الميزة الخامسة

خادم جافاسكربت المسمى (نود جى اس) يستطيع دمج اكواد (البك اند) مع (الفرونت اند) وتنفيذها دفعه واحده بسبب ان كل الاكواد المكتوب (جافاسكربت فى جافاسكربت) كلها اكواد واحده و مصدر الدعم واحد فتنفذ دفعة واحدة مما يؤدي الى زيادة السرعة فى التنفيذ وهذا مالم تستطع فعلة الخوادم الاخرى بسبب ان اكواد (البك اند) فى الخوادم الاخرى مختلفة عن (الفرونت اند) يعنى مثلا بختصار فى لغة (بى اتش بى) تكتب اكواد الباكاند بلغة (بى اتش بى) التى تعتمد فى التنفيذ على لغة (بيرل) المحملة على السيرفر و (الفرونتاند) يعتمد على ال (فى 8) المحمل ايضا على السيرفر فدمج هذه الاكواد مستحيلة لان مصادر الدعم مختلف ولا بد من كل تقنية ان تلجا للمصدر الذى يدعمها للتنفيذ انما فى سيرفر جافاسكربت الامر اكثر سهول (البك اند) جافاسكربت و (الفرونت اند) جافاسكربت ايضا تنفذ دفعة وهذه احدى الميزات التى تزيد فى سرعة سيرفر جافاسكربت (نود جى اس)

## الميزة السادسة

فى برمجة السيرفات الاخرى اذا اردت ارسال قيمة متغير من اكواد (البك اند) الى (الفرونت اند) او العكس فهذا امر معقد جدا وستحتاج الى مكتبات خاصة لتنفيذة يعنى مثلا فى لغة (بى اتش بى) ستحتاج الى مكتبة تسمى (بى اتش بى دت جى اس) لتنفيذ هذا الامر او انك ستلجا لتقنة اجاكس التى ستجبرك على انشاء ملف خاص ل(بى اتش بى) وملف خاص للجافاسكربت الذى سيرسل باكواد الاجاكس باستخدام مكتبة اخرى ستقوم بتحميلها تسمى جيكويرى اجاكس والتى سترسل البيانات الى ملف ال (بى اتش بى) الذى سيعيد ارساله مرة اخرى بعد المعالجة لتعود البيانات للملف الاول الذى اطلقت البيانات منه و هو ال (فرونت اند) كل هذه الخطواط الكثيرة و المعقدة لكى تقوم فقط بارسال قيمة متغير من اكواد البك اند الى اكواد الفرونت اند وقد تفشل العملية بعد كل هذا التعب اما فى سيرفر جافاسكربت الامر فى منتهى السهولة و اليسر فستطيع كتابت البك اند داخل الفرونت اند بلا ادنى مشكلة و(الفرونت اند) داخل (البك اند) بلا ادنى مشكله لانه لا يوجد لغة دخيلة فى السيرفر كلها لغة واحدة ستطبق دفعة واحدة بنقرة واحدة وهى جافاسكربت



## الميزة السابعة

فى برمجة السيرفات باللغات الاخرى اذا اردت انشاء موقع يعتمد على المحتوى الديناميكي من داخل الصفحة مثلا فى لغة (بى اتش بى) انشاء هذا الامر ستحتاج الى عمل الاف الاقواس المتعرجة والتي بدورها ستفصل اكواد (بى اتش بى) عن الصفحة حتى يتعرف عليها السيرفر وينفذها لانها تقنية مختلفة تعتمد على مصدر خارجى وليست من تقنية المتصفحات وهذا امر حقيقيا تنفيذه مرهق جدا وفى بعض الاحيان كثرة هذه الاقواس المتداخلة تؤدى الى حدوث ربكة عند قراءة الكود فيما بعد او تؤدى كثرتها الى اخطا فادحة ناهيك عن البطء فى التنفيذ الى جانب انه امر مزعج جدا لكثير من المطورين ولكن مع سيرفر جافاسكربت (نود جى اس) لن تحتاج الى فصل الاكواد لان كل الاكواد المكتوبه جافاسكربت لها مصدر واحد مما يسهل عليك الكتابة والتنسيق وقراءة الكود وكل شى فى صفحة واحدة

## الميزة الثامنة

هذه الميزة ستركك تعاينها بنفسك ادخل على عمو غوغل واكتب فى البحث محركات العاب جافاسكربت مفتوحة المصدر او المجانية ولن اخبرك بانك ستجد عشرات المحركات الالعاب المجانية التى ستبرهك وكلها احسن من i'h d[u، بعضها وبكفائه عالية جدا فجافاسكربت لغة قوية وسريعة فى التحريك ولذلك تفضلها معظم الشركات والمطورين المتطوعين فى صنع الالعاب حيث ان العاب جافاسكربت خفيفة جدا وسريعة و يوجد محرك العاب اسمة (جى ديفيلوب) ذووجهه عربية كاملة العبد الفقير الى الله ساهم فى تعريبة بنسة 90 بالمئة عشرات الاف من الجمل وفقنا الله فى تعربها والحمد لله رب العالمين ليصبح هذا المحرك الان ذووجهه عربية كاملة ويدعم الاخراج لكل المنصات الموبايل والكمبيوتر وكل شى مهىء للمبرمجين وغير المبرمجين فله واجهة سحب وافلات واحداث جاهزة وقوالب واشياء اخرى كثيرة انصح بتجربته وستجد العشرات من محركات الالعاب المجانية المنافسة له التى تستخدم جافاسكربت ايضا

## عيوب جافاسكربت

لن اذكر لك عيوب جافاسكربت هنا واذا اردت ان تعرف عيوب جافاسكربت اقرء الكتاب الذى سانشرة فى خلال ايام ان شاء الله تعالى اسمة الفرق بين الوبامج الهجينه والوبامج الاصلية سيشرح هذا الكتاب كيف يتم تنفيذ الوبامج الهجينه والاصلية على الكمبيوتر والفرق بينهم

استخدام محرر "جربه بنفسك"، لن نفرض عليك محرر معينة حيث ان هذه اللغة مدعومة من جميع المحررات القديمة والحديثة و المعروفة والغير معروفة حتى المحررات التى صنعتت تحت بير السلم تدعم جافاسكربت بواعة وكل المحررات تقريبا ولن تجد صعوبة فى ايجاد محرر ولكن ننصح بالفيجوال ستوديو .كود لان هو الذى سنعمل عليه فى هذا الكتاب باذن الله تعالى

## ملحوظة

نوصي بقراءة هذا الكتاب، بالتسلسل المدرج حتى يتيثر عليك الفهم  
إذا كان لديك كمبيوتر، فستكون افضل من الموبايل لانه يجب تجربة الاكواد بنفسك على الكمبيوتر  
إذا كنت تقرا من الموبايل فعليك ان تحفظ النتائج باستمرار

## فوائد التعلم من خلال الأمثلة

الأمثلة أفضل من 1000 كلمة. في الشرح لانها تعطيك النتيجة وتوفر عليك عمل تجارب كثيرة ناهيك عن الأمثلة أسهل  
في الفهم من التفسيرات النصية  
يكمل هذا الكتاب جميع الشروحات بأمثلة توضيحية ونرجوت جربها بنفسك او اكتفى بتجربة الكتاب  
إذا قمت بتجربة جميع الأمثلة، فسوف تتعلم الكثير عن الاكتفاء بتجربة الكتاب ، لانك ستجرب المثال ↓↓ بطرق كثيرة و  
! مختلفة في وقت قصير جداً ان شاء الله تعالى

## يعنى باختصار

جافاسكربت هي إحدى اللغات الثلاث التي يجب على جميع مطوري الويب تعلمها

1. لتحديد محتوى صفحات الويب HTML

2. لتحديد تخطيط صفحات الويب CSS

3. ونشاء شروط لدخال البيانات بلجافاسكربت

برمجة سلوك صفحات الويب والتحكم في الخصائص

يغطي هذا الكتاب كل إصدار من جافاسكربت: ان شاء الله تعالى

- **ES1 ES2 ES3 (1997-1999)** بدايتا من جافاسكربت الاصدار الأصلي و الاول
- **ES5 (2009)**
- **ES6 (2015)**
- اهم الإضافات تم اصدارها فى هذه السنين (2016, 2017, 2018, 2019, 2023)

ابو حبيب الحسينى

## كيف استخدام جافاسكربت فى تغير المحتوى

ضع اسم معرف اى دى للعنصر ثم استدعية `getElementById()` إحدى طرق جافاسكربت الشائعة هي

تقبل جافاسكربت علامات الاقتباس المزدوجة والمفردة

تقبل جافاسكربت علامات الاقتباس المزدوجة والمفردة

✓ مثال ↓↓

```
document.getElementById("Habib").innerHTML = "Easy-to Abu Habib Al-Husini";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

✓ مثال ↓↓

```
document.getElementById('Habib').innerHTML = 'Easy-to Abu Habib Al-Husini';
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## تغيير الخائص

:<img> للعلامة (المصدر) src في هذا ال ✓ مثال ↓↓، تغيير جافاسكربت قيمة السمة

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## كيف تغيير قيم الخائص

تغيير حجم الخط

✓ مثال ↓↓

```
document.getElementById("Habib").style.fontSize = "35px";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف اخفاء العناصر

الخاصية **display** عن طريق تغيير HTML يمكن إخفاء عناصر

✓ مثال ↓↓

```
document.getElementById("Habib").style.display = "none";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف اظهار العناصر المخفية

:الخاصية **display** المخفية عن طريق تغيير HTML يمكن أيضًا إظهار عناصر

✓ مثال ↓↓

```
document.getElementById("Habib").style.display = "block";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

هل كنت تعلم؟

جافاسكربت و جافا لغتان مختلفتان تمامًا، سواء من حيث المفهوم أو التصميم

في عام 1997 ECMA في عام 1995، وأصبح معيار Brendan Eich تم اختراع جافاسكربت بواسطة

هو الاسم الرسمي للغة ECMAScript. هو الاسم الرسمي للمعيار ECMA-262

## <script> العلامة

</script> العلامات <script> يتم إدراج كود جافاسكربت بين HTML في

مثال ↓↓ ✓

```
<script>
document.getElementById("Habib").innerHTML = "Easy-to Abu Habib Al-Husini";
</script>
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

<script type="text/javascript">. قد تستخدم أمثلة جافاسكربت القديمة سمة النوع HTML. سمة النوع غير مطلوبة. جافاسكربت هي لغة البرمجة النصية الافتراضية في

## وظائف وأحداث

"عبارة عن كتلة من تعليمات جافاسكربت البرمجية، والتي يمكن تنفيذها عند" الاستدعاء **function** جافاسكربت على سبيل ال ✓ مثال ↓↓، يمكن استدعاء دالة عند وقوع حدث ما ، مثل عندما ينقر المستخدم على زر

سوف تتعلم المزيد عن الوظائف والأحداث في الفصول اللاحقة

## كيف اضافة بلوكات من الكواد في نفس الصفحة

HTML. يمكنك وضع أي عدد من الكود في مستند

أو في كليهما HTML صفحة <head> قسم <body> يمكن وضع الكود في



## قسم الهد او راس الصفحة

HTML قسم صفحة <head> في هذا ال ✓ مثال ↓↓، يتم وضع جافاسكربت في function

:يتم استدعاء (استدعاء) الوظيفة عند النقر فوق الزر

✓ مثال ↓↓

```
<!DOCTYPE html>
<html>
<head>
<script>
function Husini() {
  document.getElementById("Habib").innerHTML = "Abu Habib Al-Husini changed.";
}
</script>
</head>
<body>

<h2>Abo Habib Al-Husini</h2>

<p id="Habib">Abo Habib Al-Husini</p>
<button type="button" onclick="Husini()">Try it</button>

</body>
</html>
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## اكتب جافاسكربت داخل الودي واستدعيها من داخل الودي

HTML قسم صفحة <body> في هذا ال ✓ مثال ↓↓، يتم وضع جافاسكربت في function

:يتم استدعاء (استدعاء) الوظيفة عند النقر فوق الزر

✓ مثال ↓↓

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

```
<body>
```

```
<h2>Abo Habib Al-Husini</h2>
```

```
<p id="Habib">Abo Habib Al-Husini</p>
```

```
<button type="button" onclick="Husini()">Try it</button>
```

```
<script>
```

```
function Husini() {
```

```
document.getElementById("Habib").innerHTML = "Abu Habib Al-Husini changed.";
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

احظر هذه الطريقة بطيئة في التنفيذ

## كتابة جافاسكريبت في ملف خارجي

يمكن أيضًا وضع الكود في ملفات خارجية وهذا هو الافضل والاسرع في التنفيذ

الملف الخارجي: **Habib\_Script.js**

```
function Husini() {
```

```
document.getElementById("Habib").innerHTML = "Abu Habib Al-Husini changed.";
```

```
}
```

تعتبر الكود الخارجية عملية عند استخدام نفس الكود في العديد من صفحات الويب المختلفة

.js . ملفات جافاسكريبت لها امتداد الملف

**<script>** للعلامة (المصدر) **src** لاستخدام برنامج نصي خارجي، ضع اسم ملف الكود في السمة

مثال ↓↓ ✓

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

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

كما تريد **<body>** أو **<head>** يمكنك وضع مرجع نصي خارجي في  
توجد فيه العلامة **<script>** سوف يتصرف الكود كما لو كان موجوداً بالضبط في المكان الذي

علامات **<script>** لا يمكن أن تحتوي الكود الخارجية على

## مزايا جافاسكربت في ملف خارجي

إن وضع الكود في ملفات خارجية له بعض المزايا

- والتعليمات البرمجية HTML فهو يفصل بين
  - وجافاسكربت أسهل في القراءة والصيانة HTML فهو يجعل
  - يمكن لملفات جافاسكربت المخزنة مؤقتاً تسريع تحميل الصفحات
- لإضافة عدة ملفات نصية إلى صفحة واحدة - استخدم عدة علامات نصية

مثال ↓↓ ✓

```
<script src="Habib_Script1.js"></script>  
<script src="Habib_Script2.js"></script>
```

## المراجع الخارجية

يمكن الرجوع إلى الكود الخارجي بثلاث طرق مختلفة

- كامل (عنوان ويب كامل) URL مع عنوان
- (مثل /js/) باستخدام مسار الملف
- دون أي طريق

Habib\_Script.js كاملاً للارتباط بـ URL يستخدم هذا ال ✓ مثال ↓↓ عنوان

مثال ↓↓ ✓

```
<script src="https://www.ab_habib_alhosiny.com/js/Habib_Script.js"></script>
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

Habib\_Script.js يستخدم هذا ال ✓ مثال ↓↓ مسار ملف للارتباط بـ

✓ مثال ↓↓

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

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

Habib\_Script.js لا يستخدم هذا ال ✓ مثال ↓↓ أي مسار للارتباط بـ

✓ مثال ↓↓

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

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

HTML. يمكنك قراءة المزيد حول مسارات الملفات في الفصل مسارات ملفات

## دوال الاخراج

### دوال تمكّنك من عرض البيانات

يمكن لجافاسكريبت "عرض" البيانات بطرق مختلفة:

- **innerHTML** باستخدام HTML الكتابة في عنصر.
- **document.write()** باستخدام HTML الكتابة في مخرجات للصفحة.
- **window.alert()** الكتابة في مربع التنبيه باستخدام.
- **console.log()** الكتابة في وحدة تحكم المتصفح باستخدام.

### HTML استخدام المحتوى الداخلي

الطريقة **document.getElementById(id)** يمكن لـ جافاسكريبت استخدام هذه HTML للوصول إلى عنصر

HTML: محتوى **innerHTML** تحدد الخاصية HTML. عنصر **id** تحدد السمة

✓ مثال ↓↓

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>Thank God The Lord Of Arabics</h1>
```

```
<p> Abo Habib Al-Husini</p>
```

```
<p id="Habib"></p>
```

```
<script>
```

```
document.getElementById("Habib").innerHTML = 5 + 6;
```

```
</script>
```

```
</body>
```

```
</html>
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

HTML. طريقة شائعة لعرض البيانات بتنسيق HTML لعنصر InternalHTML يعد تغيير خاصية

## كيف كتابة البيانات في الصفحة

write(): لأغراض الاختبار

✓ مثال ↓↓

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>Thank God The Lord Of Arabics</h1>
```

```
<p> Abo Habib Al-Husini.</p>
```

```
<script>
```

```
document.write(5 + 6);
```

```
</script>
```

```
</body>
</html>
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

: الموجود HTML إلى حذف كل HTML بعد تحميل مستند `document.write()` سيؤدي استخدام

✓ مثال ↓↓

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

```
<h1>Thank God The Lord Of Arabics</h1>
<p> Abo Habib Al-Husini.</p>
```

```
<button type="button" onclick="document.write(5 + 6)">Try it</button>
```

```
</body>
</html>
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

للاختبار فقط `document.write()` يجب استخدام التابع

## كيف اظهار رسالة للمستخدم

:يمكنك استخدام مربع التنبيه او الرساله لعرض البيانات او التوجيه الى شى ما

✓ مثال ↓↓

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

```
<h1>Thank God The Lord Of Arabics</h1>
<p> Abo Habib Al-Husini.</p>
```

```
<script>
```

```
window.alert(5 + 6);  
</script>
```

```
</body>  
</html>
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

المحجوزة **window** يمكنك تخطي الكلمة

في جافاسكربت، كائن النافذة هو كائن النطاق العام. هذا يعني أن المتغيرات والخصائص والأساليب تنتمي افتراضياً إلى كائن الكلمة المحجوزة أمر اختياري **window** النافذة. وهذا يعني أيضاً أن تحديد

✓ مثال ↓↓

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

```
<h1>Thank God The Lord Of Arabics</h1>  
<p>Abo Habib Al-Husini.</p>
```

```
<script>  
alert(5 + 6);  
</script>
```

```
</body>  
</html>
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف التسجيل داخل الكونسل

الأسلوب الموجود في المتصفح لعرض البيانات **console.log()** لأغراض تصحيح الأخطاء، يمكنك استدعاء

سوف تتعلم المزيد حول تصحيح الأخطاء في فصل لاحق

✓ مثال ↓↓

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

```
<body>
```

```
<script>
```

```
console.log(5 + 6);
```

```
</script>
```

```
</body>
```

```
</html>
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## الطباعة الخارجية في جافاسكربت

لا تحتوي جافاسكربت على أي كائن طباعة خارجية أو طرق طباعة خارجية جاهزة. لا يمكنك الوصول إلى أجهزة الإخراج من جافاسكربت إلا بمكتبات خاصة وهي كثيرة جدا وكلها تتنافس في سرعة التنفيذ. الموجودة في المتصفح لطباعة محتوى النافذة الحالية (`window.print()`) الاستثناء الوحيد هو أنه يمكنك استدعاء الطريقة

✓ مثال ↓↓

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<button onclick="window.print()">Print this page</button>
```

```
</body>
```

```
</html>
```

## تحويل البيانات في جافاسكربت

✓ مثال ↓↓

```
let x, y, z; // Statement 1
```

```
x = 5; // Statement 2
```

```
y = 6; // Statement 3
```

```
z = x + y; // Statement 4
```



## كيف تكوين البيانات البرمجية

تتكون عبارات جافاسكربت من

القيم والمعاملات والتعبيرات والكلمات المحجوزة والتعليقات

"Habib" = بمعرف HTML داخل عنصر "Abu Habib Al-Hosini" يخبر هذا البيان المتصفح بكتابة هذه الجملة

مثال ↓↓

```
document.getElementById("Habib").innerHTML = "Abu Habib Al-Husini";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

تحتوي معظم برامج جافاسكربت على العديد من عبارات جافاسكربت

يتم تنفيذ البيانات، واحدًا تلو الآخر، بنفس الترتيب الذي تمت كتابته به

غالبًا ما تسمى برامج جافاسكربت (وبيانات جافاسكربت) برمز جافاسكربت

## كيف الإعلان عن متغيرات عديدة

تفصل الفواصل المنقوطة عبارات جافاسكربت او المتغيرات

أضف فاصلة منقوطة في نهاية كل متغير او عبارة قابلة للتنفيذ

أمثلة

```
let a, b, c; // Declare 3 variables
a = 5; // Assign the value 5 to a
b = 6; // Assign the value 6 to b
c = a + b; // Assign the sum of a and b to c
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

عند الفصل بينها بفواصل منقوطة، يُسمح بعدة عبارات في سطر واحد

```
a = 5; b = 6; c = a + b;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

على الويب، قد ترى أمثلة بدون فواصل منقوطة.  
إنهاء البيانات بفاصلة منقوطة ليس مطلوبًا، ولكن يوصى به بشدة

## المسافات الفارغة

يتجاهل جافاسكربت مسافات متعددة. يمكنك إضافة مسافة فارغة إلى الكود لجعله أكثر قابلية للقراءة.  
الأسطر التالية مكافئة

```
let hAbiB = "Habib";  
let hAbiB="Habib";
```

( = + - \* / ) من الممارسات الجيدة وضع مسافات حول عوامل التشغيل

```
let x = y + z;
```

## طول السطر في جافاسكربت وفواصل الأسطر

للحصول على أفضل سهولة للقراءة، غالبًا ما يرغب المبرمجون في تجنب سطور التعليمات البرمجية التي يزيد طولها عن 80 حرفًا.

إذا كانت عبارة جافاسكربت لا تتناسب مع سطر واحد، فإن أفضل مكان لفصلها هو بعد عامل التشغيل

مثال ↓↓

```
document.getElementById("Habib").innerHTML =  
"Abu Habib Al-Hosini";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كتل الأكواد

{...} يمكن تجميع عبارات جافاسكربت معاً في كتل تعليمات برمجية، داخل قوسين متعرجين

.الغرض من كتل التعليمات البرمجية هو تحديد البيانات التي سيتم تنفيذها معاً

:أحد الأماكن التي ستجد فيها العبارات مجمعة معاً في كتل هو وظائف جافاسكربت

✓ مثال ↓↓

```
function Husini() {  
  document.getElementById("Habib1").innerHTML = "Abu Habib Al-Hosini!";  
  document.getElementById("Habib2").innerHTML = "How are you?";  
}
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

.في هذا الكتاب، نستخدم مسافتين من المسافة البادئة لكتل التعليمات البرمجية  
. سوف تتعلم المزيد عن الوظائف باذن الله تعالى في هذا الكتاب او في الجزء الثاني

## الكلمات المحجوزة في جافاسكربت

غالبًا ما تبدأ عبارات جافاسكربت بكلمة محجوزة لتحديد إجراء جافاسكربت المطلوب تنفيذه

**حمل كتاب المرجع الكامل للكلمات المحجوزة في جافاسكربت** يحتوي على جدول جميع  
الكلمات المحجوزة لجافاسكربت مع الشرح

:فيما يلي قائمة ببعض الكلمات المحجوزة التي ستتعرف عليها في هذا الكتاب

الكلمة المحجوزة	وصف
var	يعلن عن المتغير
let	يعلن عن متغير الكتلة
const	يعلن كتلة ثابتة او متغير ثابت
if	يضع علامة على كتلة من العبارات التي سيتم تنفيذها على شرط ما
switch	يحدد كتلة من البيانات التي سيتم تنفيذها في حالات مختلفة
for	يضع علامة على كتلة من العبارات التي سيتم تنفيذها في حلقة
function	تعلن عن وظيفة
return	يخرج من وظيفة

try

ينفذ معالجة الأخطاء لكثرة البيانات

الكلمات المحجوزة لجافاسكريبت هي كلمات محجوزة. لا يمكن استخدام الكلمات المحجوزة كأسماء للمتغيرات.

## طرق انشاء المتغيرات

بناء جملة جافاسكريبت او المتغيرات له طرق عديدة

// How to create variables:

```
var x;
```

```
let y;
```

// How to use variables:

```
x = 5;
```

```
y = 6;
```

```
let z = x + y;
```

## التعامل مع القيم

يحدد بناء جملة جافاسكريبت نوعين من القيم

• القيم الثابتة

• القيم المتغيرة

• تسمى القيم الثابتة بالحرفيات

• تسمى القيم المتغيرة بالمتغيرات

## التعامل مع الحروف

أهم قواعد بناء الجملة للقيم الثابتة هي

تتم كتابة الأرقام مع أو بدون الكسور العشرية .1

10.50

1001

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

النصوص النصية عبارة عن نص مكتوب بين علامتي اقتباس مزدوجتين أو مفردتين .2

"Abu Bakr Al-Siddiq"

'Abu Bakr Al-Siddiq'

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المتغيرات

في لغة الـرمجة، يتم استخدام المتغيرات لتخزين قيم البيانات

. عن المتغيرات **const** للإعلان **let** تستخدم جافاسكريبت الكلمات المحجوزة **var**

.يتم استخدام علامة المساواة لتعيين قيم للمتغيرات

:القيمة **x** كمتغير. ثم يتم تعيين (نظرًا) لـ **x** في هذا الـ **x = 6**، يتم تعريف

**let x;**

**x = 6;**

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المعاملات الحسابية

: تستخدم جافاسكريبت العوامل الحسابية ( **+** **-** **\*** **/** ) لحساب القيم

**10 \* (5 + 6)**

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تستخدم جافاسكربت عامل التعيين (=) لتعيين قيم للمتغيرات

```
let x, y;
```

```
x = 5;
```

```
y = 6;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## التعبيرات

التعبير عبارة عن مجموعة من القيم والمتغيرات وعوامل التشغيل، والتي تحسب القيمة

ويسمى الحساب التقييم

على سبيل ال مثال  $5 * 10$  يساوي 50

```
5 * 10
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يمكن أن تحتوي التعبيرات أيضًا على قيم متغيرة

```
x * 10
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يمكن أن تكون القيم من أنواع مختلفة، مثل الأرقام والنصوص

"Abu Bakr Al-Siddiq" إلى "Al Husini" + " " + "Habib" على سبيل ال مثال  $"Habib" + " " + "Al Husini"$  يتم تقييم

```
"Habib" + " " + "Al Husini"
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## الكلمات المحجوزة

تستخدم كلمات جافاسكربت المحجوزة لتحديد الإجراءات التي سيتم تنفيذها

المحجوزة المتصفح بإنشاء متغيرات `let` تخبر الكلمة

```
let x, y;  
x = 5 + 6;  
y = x * 10;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

:المحجوزة أيضًا المتصفح بإنشاء متغيرات **var** تخبر الكلمة

```
var x, y;  
x = 5 + 6;  
y = x * 10;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

.سوف يؤدي إلى نفس النتيجة **let** أو **var** في هذه الأمثلة، استخدام

.في هذا الكتاب **let** باذن الله تعالى **var** سوف تتعلم المزيد عنه

## التعليقات

.لا يتم "تنفيذ" كافة عبارات جافاسكريبت

**\*/**. يتم التعامل مع الكود بعد الشرطة المائلة المزودة **//** أو **/\*** بينهما كتعليق

:يتم تجاهل التعليقات، ولن يتم تنفيذها

```
let x = 5; // I will be executed
```

```
// x = 6; I will NOT be executed
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

.سوف تتعلم المزيد عن التعليقات في فصل لاحق

## المعرفات او الأسماء

.المعرفات هي أسماء جافاسكريبت

يتم استخدام المعرفات لتسمية المتغيرات والكلمات المحجوزة والوظائف

قواعد الأسماء القانونية هي نفسها في معظم لغات البرمجة

يجب أن يبدأ اسم جافاسكربت بـ

- حرف (من الألف إلى الياء أو من الألف إلى الياء)
- (\$) علامة الدولار
- ( \_ ) أو الشرطة السفلية

قد تكون الأحرف اللاحقة عبارة عن أحرف أو أرقام أو شرطة سفلية أو علامات الدولار

## ملحوظة

لا يسمح باستخدام الأرقام كحرف أول في الأسماء

بهذه الطريقة يمكن لـ جافاسكربت التمييز بين المعرفات والأرقام بسهولة

## احظر حساسة حالة الأحرف

جميع معرفات جافاسكربت حساسة لحالة الأحرف

عما متغيران مختلفان، `lastName` و `lastName` المتغيرات

```
let lastname, lastName;  
lastName = "Al Husini";  
lastname = "Habibson";
```

كتاب الكافي في جافاسكربت الجزء الأول ابو حبيب الحسيني

Let . على أنها الكلمة المحجوزة Let أو LET لا يفسر جافاسكربت

## تكوين الجمل

تاريخيًا، استخدم المبرمجون طرقًا مختلفة لضم كلمات متعددة في اسم متغير واحد

الواصلات

الاسم الأول، اسم العائلة، أسماء ، بين المدن يعنى اسم وصفى للشئ يعبر عنه حتى لا تتعب نفسك في كتابة تعليقات كثيرة للكود

الشرطة السفلية: او ما تعرف بالاندرسكول تستخدم لفصل الاسماء لتسهيل القراءة



الواصلات غير مسموح بها في جافاسكربت، وهي محجوزة للطرح

مثل الرموز والمعاملات الحسابية

:يميل مبرمجو جافاسكربت إلى استخدام حالة الجمل التي تبدأ بحرف صغير  
الحرف الأول، لا يكون رقم الاسم الأخير، مترك لك بحرية

## ترميز الكتابة

. Unicode يستخدم جافاسكربت مجموعة أحرف  
جميع الأحرف وعلامات الترقيم والرموز الموجودة في العالم (تقريبًا) Unicode يغطي  
. Unicode لإلقاء نظرة فاحصة، يرجى قراءة كتاب المرجع الكامل

## انشاء التعليقات

يمكن استخدام تعليقات جافاسكربت لشرح تعليمات جافاسكربت البرمجية، ولجعلها أكثر قابلية للقراءة  
يمكن أيضًا استخدام تعليقات جافاسكربت لمنع التنفيذ عند اختبار التعليمات البرمجية البديلة

## كيف انشاء تعليقات سطر واحد

// تبدأ التعليقات ذات السطر الواحد بـ  
سيتم تجاهل أي نص بين // السطر ونهايته بواسطة جافاسكربت (لن يتم تنفيذه)  
يستخدم هذا ال ✓ مثال ↓↓ تعليقًا من سطر واحد قبل كل سطر من التعليمات البرمجية

✓ مثال ↓↓

```
// Change heading:
```

```
document.getElementById("Habib1").innerHTML = "Abu Habib Al-Husini";
```

```
// Change Abo Habib Al-Husini:
```

```
document.getElementById("Habib2").innerHTML = " Abo Habib Al-Husini.";
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يستخدم هذا ال ✓ مثال ⇓⇓ تعليقا من سطر واحد في نهاية كل سطر لشرح الكود

✓ مثال ⇓⇓

```
let x = 5; // Declare x, give it the value of 5
let y = x + 2; // Declare y, give it the value of x + 2
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

### كيف انشاء تعليقات متعددة الأسطر

/\* التعليقات متعددة الأسطر تبدأ ب / \* وتنتهي ب

. سيتم تجاهل أي نص بين / \* و / \* سيتم تجاهله بواسطة جافاسكربت

يستخدم هذا ال ✓ مثال ⇓⇓ تعليقا متعدد الأسطر (كتلة تعليق) لشرح الكود

✓ مثال ⇓⇓

```
/*
The code below will change
the heading with id = "Habib1"
and the Abo Habib Al-Husini with id = "Habib2"
in my web page:
*/
document.getElementById("Habib1").innerHTML = "Abu Habib Al-Husini";
document.getElementById("Habib2").innerHTML = "Abo Habib Al-Husini.";
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

من الأكثر شيوعاً استخدام التعليقات ذات السطر الواحد  
غالباً ما تُستخدم التعليقات الجماعية للتوثيق الرسمي

### استخدام التعليقات لمنع التنفيذ

يعد استخدام التعليقات لمنع تنفيذ التعليمات البرمجية مناسباً لاختبار التعليمات البرمجية

تؤدي الإضافة // أمام سطر التعليمات البرمجية إلى تغيير أسطر التعليمات البرمجية من سطر قابل للتنفيذ إلى تعليق

يستخدم هذا ال ✓ مثال ↓↓ // لمنع تنفيذ أحد أسطر التعليمات البرمجية

✓ مثال ↓↓

```
//document.getElementById("Habib1").innerHTML = "Abu Habib Al-Husini";  
document.getElementById("Habib2").innerHTML = "Abo Habib Al-Husini.";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يستخدم هذا ال ✓ مثال ↓↓ كتلة تعليق لمنع تنفيذ أسطر متعددة

✓ مثال ↓↓

```
/*  
document.getElementById("Habib1").innerHTML = "Abu Habib Al-Husini";  
document.getElementById("Habib2").innerHTML = "Abo Habib Al-Husini.";  
*/
```

## متغيرات جافاسكربت

طرق للإعلان عن متغير جافاسكربت 4

- استخدام var
- استخدام let
- استخدام const
- عدم الاستخدام أي شيء تستطيع إعلان عن متغير

## ما هي المتغيرات؟

المتغيرات عبارة عن حاويات لتخزين البيانات (تخزين قيم البيانات)

:المحجوزة **var** هي متغيرات تم الإعلان عنها باستخدام الكلمة **z** و **y** و **x**، في هذا ال مثال

مثال

```
var x = 5;  
var y = 6;  
var z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

**let** مثال على استخدام الكلمة

مثال

```
let x = 5;  
let y = 6;  
let z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

:هي متغيرات غير معلنة **z** و **y** و **x**، في هذا ال مثال

مثال

```
x = 5;  
y = 6;  
z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

من جميع الأمثلة المذكورة أعلاه، يمكنك تخمين: نوع المتغير بمجرد تخزين القيم في الذاكرة

- يخزن القيمة **5** **x**
- يخزن القيمة **6** **y**
- يخزن القيمة **11** **z**

## متى استخدام الكلمة var ؟

**const** أو **let** أو **var** أعلن دائماً عن متغيرات جافاسكربت باستخدام

. يتم استخدام الكلمة المحجوزة في جميع أكواد جافاسكربت من عام 1995 إلى عام 2015 **var**

**let.const** تمت إضافة الكلمات المحجوزة والكلمات المحجوزة إلى جافاسكربت في عام 2015

**var**. إذا كنت تريد تشغيل التعليمات البرمجية في المتصفحات القديمة، فيجب عليك استخدام

## متى استخدام الكلمة const ؟

**const**. إذا كنت تريد قاعدة عامة: أعلن دائماً عن المتغيرات باستخدام

**let**. إذا كنت تعتقد أن قيمة المتغير يمكن أن تتغير، فاستخدم

:هي متغيرات **total** و **price2** و **price1**، **price1** و **price2** في هذا المثال ✓

✓ مثال ↓↓

```
const price1 = 5;  
const price2 = 6;  
let total = price1 + price2;
```

كتاب الكافي في جافاسكربت الجزء الأول أبو حبيب الحسيني

الكلمة المحجوزة **const** باستخدام **price2** يتم الإعلان عن المتغيرين **price1**

هذه قيم ثابتة ولا يمكن تغييرها

الكلمة المحجوزة **let** يتم الإعلان عن المتغير باستخدام **total**

هذه هي القيمة التي يمكن تغييرها

## استخدام معملات الجبر

:كما هو الحال في الجبر، تحتوي المتغيرات على قيم

let x = 5;

let y = 6;

كما هو الحال في الجبر، يتم استخدام المتغيرات في التعبيرات

let z = x + y;

من ال ✓ مثال ↓↓ أعلاه، يمكنك تخمين أن الإجمالي قد تم حسابه ليكون 11

## ملحوظة

المتغيرات هي حاويات لتخزين القيم.

## ما هي معرفات جافاسكربت

• يجب تعريف كافة متغيرات جافاسكربت بأسماء فريدة

• تسمى هذه الأسماء الفريدة المعرفات

• أو أسماء وصفية أكثر (العمر، المجموع، الحجم الإجمالي) (y و x مثل) يمكن أن تكون المعرفات أسماء قصيرة

القواعد العامة لبناء أسماء المتغيرات (المعرفات الفريدة) هي:

- يمكن أن تحتوي الأسماء على أحرف وأرقام وشرطات سفلية وعلامات الدولار.
- يجب أن تبدأ الأسماء بحرف.
- يمكن أن تبدأ الأسماء أيضًا بـ \$ و (لكننا لن نستخدمها في هذا الكتاب ولا ننصح باستخدامها لأنها مستخدمة في مكتبات أخرى مثل الجيكويري وغيرها والتي سنتطرق إلى إليها بالتفصيل أما في هذا الكتاب أو في كتب أخرى).
- (متغيران مختلفان y و Y) الأسماء حساسة لحالة الأحرف.
- لا يمكن استخدام الكلمات المحجوزة (مثل الكلمات المحجوزة لـ جافاسكربت) كأسماء.

## ملحوظة

معرفات جافاسكربت حساسة لحالة الأحرف.

## عامل التعيين

"في جافاسكربت، علامة المساواة (=) هي عامل تشغيل "إسناد"، وليس عامل تشغيل "يساوي"

وهذا يختلف عن الجبر. سيحددها المتصفح بعد تخمين نوع البيانات او المتغيرات و ما يلي لا معنى له في الجبر

**x = x + 5**

x إلى x + 5 ومع ذلك، في جافاسكربت، يكون الأمر منطقيًا تمامًا: فهو يعين قيمة (بمقدار x 5 وتزداد قيمة x ويضع النتيجة في x + 5 بحسب قيمة)

## ملحوظة

تتم كتابة عامل التشغيل "يساوي" كما هو الحال == في جافاسكربت

## أنواع البيانات السائدة

"Abu Bakr Al-Siddiq" يمكن لمتغيرات جافاسكربت أن تحتوي على أرقام مثل 100 وقيم نصية مثل في البرمجة، تسمى القيم النصية نصوص نصية. يمكن لجافاسكربت التعامل مع العديد من أنواع البيانات، ولكن في الوقت الحالي، فكر فقط في الأرقام والنصوص. تتم كتابة النصوص داخل علامات اقتباس مزدوجة أو مفردة. تتم كتابة الأرقام دون علامات الاقتباس. إذا وضعت رقمًا بين علامتي اقتباس، فسيتم التعامل معه كنصوص.

مثال ↓↓ ✓

```
const pi = 3.14;  
let hAbiB = "Abu Bakr Al-Siddiq";  
let Text = 'Yes I am!';
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف الإعلان عن متغير

يُسمى إنشاء متغير في جافاسكربت "الإعلان" عن متغير

let: أو var قمت بتعريف متغير جافاسكربت باستخدام الكلمة المحجوزة

**var hAbiB4;**

أو:

```
let hAbiB4;
```

(**undefined** من الناحية الفنية هو) بعد التصريح، ليس للمتغير أي قيمة.

: لتعيين قيمة للمتغير، استخدم علامة التساوي

```
hAbiB4 = "Hamza";
```

:يمكنك أيضًا تعيين قيمة للمتغير عندما تعلن عنه

```
let hAbiB4 = "Hamza";
```

له "Hamza" وتخصيص القيمة **hAbiB4** في ال **hAbiB4** مثال **hAbiB4** أدناه. قمنا بإنشاء متغير يسمى

**id="Habib"** باستخدام **HTML** ثم نقوم "بإخراج" القيمة داخل فقرة

مثال **hAbiB4**

```
<p id="Habib"></p>
```

```
<script>
```

```
let hAbiB4 = "Hamza";
```

```
document.getElementById("Habib").innerHTML = hAbiB4;
```

```
</script>
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ملحوظة

. جيد ان تقوم بـ للإعلان عن جميع المتغيرات في بداية الكود لتسهيل الرجوع اليها مرة اخرى

## كيف عمل اعلان واحد، لجميع المتغيرات

.يمكنك إعلان العديد من المتغيرات في عبارة واحدة

: المتغيرات **let** ابدأ العبارة بالفاصلة بين



✓ مثال ↓↓

```
let hAbiB = "Abu Bakr Al-Siddiq", hAbiB2= "Hamza", hAbiB3= 200;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يمكن أن يمتد الإعلان على عدة أسطر

✓ مثال ↓↓

```
let hAbiB = "Abu Bakr Al-Siddiq",  
hAbiB4 = "Hamza",  
hAbiB3= 200;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف جعل القيمة غير محدد

في برامج الكمبيوتر، غالبًا ما يتم الإعلان عن المتغيرات بدون قيمة. يمكن أن تكون القيمة شيئًا يجب حسابه، أو شيئًا سيتم توفيره بأذن الله تعالى، مثل إدخال المستخدم.

**undefined** المتغير المعلن بدون قيمة سيكون له القيمة.

بعد تنفيذ هذا البيان **undefined** القيمة **hAbiB2** سيكون للمتغير

✓ مثال ↓↓

```
let hAbiB2=undefined ;
```

```
let hAbiB4;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## إعادة الإعلان عن متغيرات

فلن يفقد قيمته، **var** إذا قمت بإعادة تعريف متغير جافاسكربت الذي تم تعريفه بـ

بعد تنفيذ هذه العبارات "Hamza" يحمل القيمة **hAbiB4** سيظل المتغير

✓ مثال ↓↓

```
var hAbiB4 = "Hamza";  
var hAbiB4;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة

**const** أو **let** لا يمكنك إعادة تعريف متغير تم تعريفه بـ

هذا لن يعمل:

```
let hAbiB4 = "Hamza";  
let hAbiB4;
```

## انشاء العمليات الحسابية

كما هو الحال مع الجبر، يمكنك إجراء العمليات الحسابية باستخدام متغيرات جافاسكربت، باستخدام عوامل تشغيل **+** مثل **=** و

✓ مثال ↓↓

```
let x = 5 + 2 + 3;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يمكنك أيضًا إضافة نصوص، ولكن سيتم ربط النصوص

✓ مثال ↓↓

```
let x = "Habib" + " " + "Al Husini 🇸🇦🇸🇦🇸🇦";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

جرب هذا أيضًا

✓ مثال ↓↓

```
let x = "5" + 2 + 3;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة

إذا وضعت رقمًا بين علامتي اقتباس، فسيتم التعامل مع بقية الأرقام كنصوص، وستكون متنص

:الآن جرب هذا

✓ مثال ↓↓

```
let x = 2 + 3 + "5";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## \$ استخدام علامة الدولار

نظرًا لأن جافاسكربت تتعامل مع علامة الدولار كحرف، فإن المعرفات التي تحتوي على \$ هي أسماء متغيرات صالحة

✓ مثال ↓↓

```
let $ = "Abu Habib Al Husini *_*";
```

```
let $$$ = 2;
```

```
let $Habib = 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

استخدام علامة الدولار ليس شائعًا جدًا في جافاسكربت، لكن غالبًا ما يستخدمها المبرمجون في بي اتش بي لانهم تعودو عليها وهي اسلوب احترام لكونها مستخدمة في مكتبات كثيرة بهذا الشكل يستخدمها المحترفون كاسم مستعار للوظيفة المحجوزة في مكتبة جيكويري جافاسكربت.

على سبيل ال ✓ مثال ↓↓، يتم استخدام الوظيفة المحجوزة \$ لتحديد عناصر jQuery في مكتبة جافاسكربت على سبيل ال ✓ مثال ↓↓، يتم استخدام الوظيفة المحجوزة \$ لتحديد عناصر jQuery في HTML. "p" تعني "تحديد جميع عناصر jQuery \$("p")".

## جافاسكريت و الاندرسكول

نظراً لأن جافاسكريت تتعامل مع الشرطية السفلية المعروفة بالاندرسكول كحرف، فإن المعارف التي تحتوي على \_ هي أسماء متغيرات صالحة:

مثال ↓↓ ✓

```
let _Name= "tariq ibn Ziad";  
let _x = 2;  
let _100 = 5;
```

كتاب الكافي في جافاسكريت الجزء الاول ابو حبيب الحسيني

استخدام الشرطية السفلية ليس شائعاً جداً في جافاسكريت، ولكن هناك تقليد بين المبرمجين المحترفين هو استخدامها كاسم "مستعار للمتغيرات" الخاصة (المخفية).

### ملحوظة

ES6 (2015) تم تقديم الكلمة المحجورة في **let**.

لا يمكن إعادة تعريف المتغيرات المحددة **let**.

يجب الإعلان عن المتغيرات المحددة قبل الاستخدام **let**.

نطاق كتلة **let** المتغيرات المحددة لها.

### احظر هذا الخطا عند استخدام الكلمة Let

لا يمكن إعادة تعريف المتغيرات المعرفة بـ **let**.

لا يمكنك إعادة تعريف متغير عن طريق الخطأ.

أنت لا تستطيع أن تفعل هذا **let** مع

✓ مثال ↓↓

```
let x = "Abu Bakr Al-Siddiq";  
let x = 0;  
// SyntaxError: 'x' has already been declared
```

يمكن **var** معك ان يفعلها

✓ مثال ↓↓

```
var x = "Abu Bakr Al-Siddiq";  
var x = 0;
```

## ما هو نطاق الكتلة { }

. كان لجافاسكربت نطاق عام ونطاق وظيفي فقط، (2015) ES6 قبل

**let** و **const**: كلمتين رئيسيتين جديدتين مهمتين في جافاسكربت ES6 قدم

. توفر هاتان الكلمتان الرئسيتان نطاق الكتلة في جافاسكربت

:لا يمكن الوصول إلى المتغيرات المعلنة داخل الكتلة { } من خارج الكتلة

✓ مثال ↓↓

```
{  
  let x = 2;  
}  
// x can NOT be used here
```

.الكلمة المحجوزة أن تحتوي على نطاق حظر **var** لا يمكن للمتغيرات المعلنة باستخدام

.يمكن الوصول إلى المتغيرات المعلنة داخل الكتلة { } من خارج الكتلة

✓ مثال ↓↓

```
{  
  var x = 2;  
}  
// x CAN be used here
```

## إعادة الإعلان عن المتغيرات

الكلمة المحجوزة إلى حدوث مشكلات **var** يمكن أن تؤدي إعادة الإعلان عن متغير باستخدام إعادة تعريف المتغير داخل الكتلة سيؤدي أيضًا إلى إعادة تعريف المتغير خارج الكتلة:

✓ مثال ↓↓

```
var x = 10;  
// Here x is 10  
  
{  
  var x = 2;  
  // Here x is 2  
}  
  
// Here x is 2
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

الكلمة المحجوزة إلى حل هذه المشكلة **let** يمكن أن تؤدي إعادة تعريف المتغير باستخدام إعادة تعريف المتغير داخل الكتلة لن يعيد تعريف المتغير خارج الكتلة:

✓ مثال ↓↓

```
let x = 10;  
// Here x is 10  
  
{  
  let x = 2;  
  // Here x is 2  
}  
  
// Here x is 10
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## إعادة الإعلان للمتغيرات

يُسمح بإعادة الإعلان عن متغير جافاسكربت في أي مكان في البرنامج **var**:

## ✓ مثال ↓↓

```
var x = 2;  
// Now x is 2  
  
var x = 3;  
// Now x is 3
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

لا يُسمح بإعادة تعريف متغير في نفس الكتلة **let** مع

## ✓ مثال ↓↓

```
var x = 2; // Allowed  
let x = 3; // Not allowed  
  
{  
  let x = 2; // Allowed  
  let x = 3; // Not allowed  
}  
  
{  
  let x = 2; // Allowed  
  var x = 3; // Not allowed  
}
```

مسموح به في كتلة أخرى **let** إعادة تعريف متغير باستخدام

## ✓ مثال ↓↓

```
let x = 2; // Allowed  
  
{  
  let x = 3; // Allowed  
}  
  
{  
  let x = 4; // Allowed  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف استخدام المتغيرات قبل الإعلان عنها

لا انصح باستخدام هذه الطريقة لانها تحتم عليك اعطاء امر لجافاسكربت بقراءة الصفحة كاملة قل التنفيذ مما يتيح لك استخدام المتغيرات قبل الاعلان عنها وهذه الطريقة تؤدي الى ارهاق البيروسييسور اقاء كتاب تعلم برمجة جيكويري الجزء الثانى لتعرف المزيد عن التحكم فى قراءة صفحات الويب والتحكم فى البيانات بمكتبة جيكويري

يتم رفع المتغيرات المحددة إلى الأعلى ويمكن تهيئتها في أي وقت **var**

:المعنى: يمكنك استخدام المتغير قبل الإعلان عنه

✓ مثال ↓↓

:لابأس

```
hAbiB4 = "Hamza";
```

```
var hAbiB4;
```

كتاب الكافي فى جافاسكربت الجزء الاول ابو حبيب الحسينى

إذا كنت تريد معرفة المزيد عن الرفع، فقرأ كتاب **جافاسكربت هوستنج** .

**let** ملحوظة لا تفعل هذا مع ليت

: المتغير قبل الإعلان عنه سيؤدي الى خط مع **let** الا تستخدم :

✓ مثال ↓↓

```
hAbiB4 = "Habib";
```

```
let hAbiB4 = "Hamza";
```

كتاب الكافي فى جافاسكربت الجزء الاول ابو حبيب الحسينى

## استخدام الكلمة const

. (2015) ES6 تم تقديم الكلمة المحجوزة في **const**

. لا يمكن إعادة تعريف المتغيرات المحددة **const**



لا يمكن إعادة تعيين المتغيرات المحددة **const**.

نطاق كتلة **const** المتغيرات المحددة لها.

## خصائص الكلمة كونسنت

: لا يمكن إعادة تعيين المتغير مرة أخرى إذا تم الاعلان عنه بكونست **const**

✓ مثال ↓↓

```
const PI = 3.141592653589793;  
PI = 3.14; // This will give an error  
PI = PI + 10; // This will also give an error
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## يجب أن يتم تعيينه

: يجب تعيين قيمة لمتغيرات جافاسكربت عند الإعلان عنها **const**

صحيح

```
const PI = 3.14159265359;
```

غير صحيح

```
const PI;  
PI = 3.14159265359;
```

## const متى تستخدم

عندما تعلم أنه لا ينبغي تغيير القيمة **const** أعلن دائماً عن متغير

:عندما تعلن **const** استخدم

- مصفوفة جديدة
- كائن جديد
- وظيفة جديدة
- جديد RegExp

## الكائنات والمصفوفات

مضللة بعض الشيء، **const** الكلمة المحجوزة  
ولا يحدد قيمة ثابتة. يحدد مرجحًا ثابتًا للقيمة  
ولهذا السبب لا يمكنك:

- إعادة تعيين قيمة ثابتة
- إعادة تعيين مجموعة ثابتة
- إعادة تعيين كائن ثابت

؛ ولكن يمكنك:

- تغيير عناصر المصفوفة الثابتة
- تغيير خصائص الكائن الثابت

## تطبيق عملي على المصفوفات

يمكنك تغيير عناصر مصفوفة ثابتة

✓ مثال ↓↓

```
// You can create a constant array:
```

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

```
// You can change an element:
```

```
HaBiB[0] = "Abu Bakr Al-Siddiq!!!!";
```

```
// You can add an element:
```

```
HaBiB.push("Omar ibn Al-khattab");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

لكن لا يمكنك إعادة تعيين المصفوفة

✓ مثال ↓↓

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

```
HaBiB= ["Abu Bakr Al-Siddiq!!!", "Hamza", "Omar ibn Al-khattab"]; // ERROR
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## مثال اخر على الكائنات

يمكنك تغيير خصائص كائن ثابت

✓ مثال ↓↓

```
// You can create a const object:
```

```
const $_Habib= {use:"Mohamed", age:"20", color:"white"};
```

```
// You can change a property:
```

```
$_Habib.color = "red";
```

```
// You can add a property:
```

```
$_Habib.use = "tariq ibn Ziad";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

لكن لا يمكنك إعادة تعيين الكائن

✓ مثال ↓↓

```
const $_Habib= {use:"Mohamed", age:"20", color:"white"};
```

```
$_Habib= {use:"Hamza", age:"EX60", color:"red"}; // ERROR
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## نطاق المتغير داخل الكتلة

Block Scope عندما يتعلق الأمر بـ **let** يشبه ما يحدث **const** الإعلان عن متغير

:المعلنة خارج الكتلة X المعلنة في الكتلة. في هذا ال ✓ مثال ↓↓، ليست هي نفس X إن

✓ مثال ↓↓

```
const x = 10;  
// Here x is 10  
  
{  
  const x = 2;  
  // Here x is 2  
}  
  
// Here x is 10
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف إعادة الإعلان عن المتغير

يُسمح بإعادة الإعلان عن متغير جافاسكربت في أي مكان في البرنامج **var**:

✓ مثال ↓↓

```
var x = 2; // Allowed  
var x = 3; // Allowed  
x = 4; // Allowed
```

إلى نفس النطاق **const** متغير **let** أو **var** لا يُسمح بإعادة الإعلان عن موجود

✓ مثال ↓↓

```
var x = 2; // Allowed  
const x = 2; // Not allowed
```

```
{  
  let x = 2; // Allowed  
  const x = 2; // Not allowed  
}
```

```
{  
  const x = 2; // Allowed  
  const x = 2; // Not allowed  
}
```

في نفس النطاق، **const** لا يُسمح بإعادة تعيين متغير موجود

✓ مثال ↓↓

```
const x = 2; // Allowed
x = 2;      // Not allowed
var x = 2;  // Not allowed
let x = 2;  // Not allowed
const x = 2; // Not allowed

{
  const x = 2; // Allowed
  x = 2;      // Not allowed
  var x = 2;  // Not allowed
  let x = 2;  // Not allowed
  const x = 2; // Not allowed
}
```

يُسمح بإعادة تعريف متغير باستخدام ، في نطاق آخر، أو في كتلة أخرى **const**

✓ مثال ↓↓

```
const x = 2; // Allowed

{
  const x = 3; // Allowed
}

{
  const x = 4; // Allowed
}
```

## المزيد عن الرفع وإعادة الإعلان

يتم رفع المتغيرات المحددة إلى الأعلى ويمكن تهيئتها في أي وقت **var**

:المعنى: يمكنك استخدام المتغير قبل الإعلان عنه

✓ مثال ↓↓

تلايس

```
hAbiB4 = "Hamza";  
var hAbiB4;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

إذا كنت تريد معرفة المزيد عن الرفع، فقرأ كتاب المرجع الكامل للجافاسكريبت  
بإلى الأعلى، ولكن لا تتم تهيئتها **const** يتم أيضًا رفع المتغيرات المعرفة  
**ReferenceError** المتغير قبل الإعلان عنه سيؤدي إلى **const** المعنى: استخدام

مثال ↓↓ ✓

```
alert (hAbiB4);  
const hAbiB4 = "Hamza";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## طرق تعيين القيمة

يقوم عامل التعيين (=) بتعيين قيمة لمتغير

أمثلة

```
let x = 10;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

```
// Assign the value 5 to x  
let x = 5;  
// Assign the value 2 to y  
let y = 2;  
// Assign the value x + y to z:  
let z = x + y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

يقوم عامل الإضافة (+) بإضافة الأرقام

## إضافة

```
let x = 5;  
let y = 2;  
let z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

عامل الضرب (\*) يقوم بضرب الأرقام

## ضرب

```
let x = 5;  
let y = 2;  
let z = x * y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## أنواع معاملات جافاسكربت

هناك أنواع مختلفة من عوامل تشغيل جافاسكربت

- العمليات الحسابية
- معاملات التعيين
- عوامل المقارنة
- العوامل المنطقية
- العوامل الشرطية
- نوع المعاملات ن

## معاملات جافاسكربت الحسابية

يتم استخدام العوامل الحسابية لإجراء العمليات الحسابية على الأرقام

## أمثلة على العوامل الحسابية

```
let a = 3;  
let x = (100 + 50) * a;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

Operator	وصف
+	إضافة
-	الطرح
*	عملية الضرب
**	الأسّي (ES2016)
/	قسم
%	المعامل (باقي القسمة)
++	زيادة راتب
--	إنقاص

## معاملات تعيين جافاسكربت

يقوم مشغلو التعيين بتعيين قيم لمتغيرات جافاسكربت

يضيف عامل تعيين الإضافة (+) قيمة إلى متغير

### تكليف

```
let x = 10;
x += 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

Operator	مثال	Same As
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x %= y$	$x = x \% y$
**=	$x **= y$	$x = x ** y$



## إضافة ودمج النصوص في جافاسكريبت

يمكن أيضًا استخدام عامل + التشغيل لإضافة (تسلسل) النصوص

✓ مثال ↓↓

```
let text1 = "Habib";  
let text2 = "Al Husini 🍌🍌🍌";  
let text3 = text1 + " " + text2;
```

نتيجة النص 3 ستكون

Abu Bakr Al-Siddiq

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

يمكن أيضًا استخدام عامل += المهمة لإضافة (تسلسل) النصوص

✓ مثال ↓↓

```
let text1 = "What a very ";  
text1 += "nice day";
```

نتيجة النص 1 ستكون

What a very nice day

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

عند استخدامه على النصوص، يُسمى عامل التشغيل + عامل التسلسل

## إضافة نصوص وأرقام

: تؤدي إضافة رقمين إلى إرجاع المجموع، لكن إضافة رقم ونص سيؤدي إلى إرجاع نص

✓ مثال ↓↓

```
let x = 5 + 5;  
let y = "5" + 5;  
let z = "Easy-to" + 5;
```

ستكون Z و Y و X نتيجة

10  
55  
Easy-to5

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

! إذا قمت بإضافة رقم ونص ، ستكون النتيجة نص

## معاملات المقارنة

المعامل	وصف
==	يساوي
===	قيمة متساوية ونوع متساو
!=	غير متساوي
!==	قيمة غير متساوية أو نوع غير متساو
>	أكثر من
<	أقل من
>=	أكبر من أو يساوي
<=	أقل أو يساوي
?	عامل ثلاثي

.JS تم وصف عوامل المقارنة بشكل كامل في فصل **مقارنات**

## عوامل المنطقية في ووضغ الشروط

Operato ر	Description
&&	و
	او
!	ليس

.JS تم وصف العوامل المنطقية بشكل كامل في فصل **المقارنات**

## معاملات الأنواع

typeof	إرجاع نوع المتغير
instanceof	يُرجع صحيحًا إذا كان الكائن مئيلاً لنوع الكائن

.JS تم وصف عوامل تشغيل الكتابة بشكل كامل في فصل **تحويل نوع**

## معاملات Bitwise

يعمل مشغلو البت على أرقام 32 بت

يتم تحويل أي معامل رقمي في العملية إلى رقم 32 بت. يتم تحويل النتيجة مرة أخرى إلى رقم جافاسكربت

Operato ر	وصف	مثال	Same as	Result	Decimal
&	و	5 & 1	0101 & 0001	0001	1
	أو	5   1	0101   0001	0101	5
~	لا	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	التحول الأيسر	5 << 1	0101 << 1	1010	10



✓ مثال ↓↓

let x = 100 + 50;

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

أو المتغيرات

✓ مثال ↓↓

let x = a + b;

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

أو التعبيرات

✓ مثال ↓↓

let x = (100 + 50) \* a;

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ما هي المعاملات

• تسمى الأرقام (في العمليات الحسابية) بالمعاملات

• يتم تعريف العملية (التي سيتم تنفيذها بين المعاملين) بواسطة عامل التشغيل

المشغل أو العامل المعامل	المعامل
100 +	50

## كيف الاستخدام في عملية الجمع

✓ مثال ↓↓

let x = 5;  
let y = 2;  
let z = x + y;

## الطرح

يقوم عامل الطرح (-) بطرح الأرقام

✓ مثال ↓↓

```
let x = 5;  
let y = 2;  
let z = x - y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

## الضرب

عامل الضرب (\*) يقوم بضرب الأرقام

✓ مثال ↓↓

```
let x = 5;  
let y = 2;  
let z = x * y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

## القسمة

عامل القسمة (/) يقسم الأرقام

✓ مثال ↓↓

```
let x = 5;  
let y = 2;  
let z = x / y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

## بقية القسمة

يقوم عامل المعامل (%) بإرجاع باقي القسمة.

✓ مثال ↓↓

```
let x = 5;  
let y = 2;  
let z = x % y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

. في الحساب، قسمة عددين صحيحين ينتج خارج القسمة والباقي

. في الرياضيات، نتيجة العملية المعيارية هي باقي القسمة الحسابية

## الجمع بالزيادة الثابتة

.عامل الزيادة (++) يزيد الأرقام بواحد في كل اجراء

✓ مثال ↓↓

```
let x = 5;  
x++;  
let z = x;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

## الطرح

.عامل التناقص (--) ينقص الأرقام

✓ مثال ↓↓

```
let x = 5;  
x--;  
let z = x;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## الأس

يقوم عامل الأس (\*\*) برفع المعامل الأول إلى أس المعامل الثاني

✓ مثال ↓↓

```
let x = 5;  
let z = x ** 2;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

**Math.pow(x,y):** ينتج نفس النتيجة مثل  $x ** y$

✓ مثال ↓↓

```
let x = 5;  
let z = Math.pow(x,2);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## أسبقية المشغل في التنفيذ

تصف أسبقية عامل التشغيل الترتيب الذي يتم به تنفيذ العمليات في التعبير الحسابي

✓ مثال ↓↓

```
let x = 100 + 50 * 3;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

هل نتيجة ال ✓ مثال ↓↓ أعلاه هي نفسها  $3 * 150$  أم هي نفسها  $100 + 150$ ؟



هل يتم الجمع أم الضرب أولاً؟

كما هو الحال في الرياضيات المدرسية التقليدية، يتم الضرب أولاً

و(+) الضرب (\*) والقسمة (/) لهما أسبقية أعلى من الجمع (+) والطرح

و(كما هو الحال في الرياضيات المدرسية) يمكن تغيير الأسبقية باستخدام الأقواس

:عند استخدام الأقواس، يتم حساب العمليات داخل الأقواس أولاً

✓ مثال ↓↓

$$\text{let } x = (100 + 50) * 3;$$

كتاب الكافي في جافاسكريبت الجزء الأول ابو حبيب الحسيني

:عندما يكون للعديد من العمليات نفس الأسبقية (مثل الجمع والطرح أو الضرب والقسمة)، يتم حسابها من اليسار إلى اليمين

أمثلة

$$\text{let } x = 100 + 50 - 3;$$

كتاب الكافي في جافاسكريبت الجزء الأول ابو حبيب الحسيني

$$\text{let } x = 100 / 50 * 3;$$

كتاب الكافي في جافاسكريبت الجزء الأول ابو حبيب الحسيني

## ملحوظة

:للحصول على قائمة كاملة بقيم أسبقية عامل التشغيل، انتقل إلى

. [قيم أسبقية مشغل جافاسكريبت](#)

# تعرف على المشغلات بالأمثلة

## مشغلات التعيين في جافاسكربت

يقوم مشغلو التعيين بتعيين قيم لمتغيرات جافاسكربت.

Operator	مثال	Same As
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$
**=	$x ** = y$	$x = x ** y$

## مشغلي مهمة التحول

Operator	مثال	Same As
<<=	$x \ll = y$	$x = x \ll y$
>>=	$x \gg = y$	$x = x \gg y$
>>>=	$x \ggg = y$	$x = x \ggg y$

## Bitwise مشغلي تعيين

Operator	مثال	Same As
&=	$x \& = y$	$x = x \& y$
^=	$x \wedge = y$	$x = x \wedge y$
=	$x   = y$	$x = x   y$

## مشغلي التعيين المنطقي

Operator	مثال	Same As
&&=	$x \&\& = y$	$x = x \&\& (x = y)$
=	$x    = y$	$x = x    (x = y)$
??=	$x ?? = y$	$x = x ?? (x = y)$

= المشغل

يقوم عامل التعيين البسيط بتعيين قيمة لمتغير.

## أمثلة مهمة بسيطة

```
let x = 10;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

```
let x = 10 + y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## += المشغل

يضيف عامل تعيين الإضافة قيمة إلى متغير

## إضافة أمثلة التعيين

```
let x = 10;
```

```
x += 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

```
let text = "Easy-to"; text += " Arabic";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## -= المشغل

يقوم عامل تعيين الطرح بطرح قيمة من متغير

مثال ↓↓ على مهمة الطرح ✓

```
let x = 10;
```

```
x -= 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المشغل = \*

يقوم عامل تعيين الضرب بضرب متغير

مثال  $\Downarrow\Downarrow$  على مهمة الضرب ✓

```
let x = 10;  
x *= 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المشغل = \*\*

يقوم عامل التعيين الأسّي برفع متغير إلى قوة المعامل

مثال  $\Downarrow\Downarrow$  على التنازل الأسّي ✓

```
let x = 10;  
x **= 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المشغل /=

يقوم عامل تخصيص القسم بتقسيم المتغير

مثال  $\Downarrow\Downarrow$  لتخصيص القسم ✓

```
let x = 10;  
x /= 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المشغل %

يقوم عامل تعيين الباقيمن القسمة بتعيين الباقي لمتغير

مثال  $\Downarrow\Downarrow$  لتعيين الباقي ✓

```
let x = 10;  
x %= 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## $\ll=$ العامل

الأيسر بإزاحة متغير Shift يقوم عامل تعيين

مثال  $\Downarrow\Downarrow$  لتعيين التحول الأيسر ✓

```
let x = -100;  
x <<= 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## $\gg=$ المشغل

الأيمن بإزاحة متغير (الموقع) Shift يقوم عامل تعيين

مثال  $\Downarrow\Downarrow$  لتعيين التحول الأيمن ✓

```
let x = -100;  
x >>= 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## $\ggg=$ المشغل

يقوم عامل تعيين التحول الأيمن غير الموقع بإزاحة متغير (غير موقع) إلى اليمين

مثال  $\Downarrow\Downarrow$  لتعيين التحويل الأيمن غير الموقع ✓

```
let x = -100;  
x >>>= 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## المشغل &=

على معاملين ويقوم بتعيين النتيجة للمتغير AND بإجراء عملية Bitwise AND Assignment يقوم عامل

مثال  $\Downarrow\Downarrow$  للبت والتخصيص ✓

```
let x = 10;  
x &= 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## المشغل |=

على معاملين ويقوم بتعيين النتيجة للمتغير bitwise OR بإجراء عملية Bitwise OR Assignment يقوم عامل

أو الواجب Bitwise  $\Downarrow\Downarrow$  مثال ✓

```
let x = 10;  
x |= 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## المشغل ^=

على معاملين ويقوم بتعيين النتيجة للمتغير XOR بإجراء عملية Bitwise XOR يقوم عامل تعيين

## ✓ لتعيين $\Downarrow\Downarrow$ مثال Bitwise XOR

```
let x = 10;  
x ^= 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## == && المشغل

بين قيمتين AND يتم استخدام عامل التشغيل المنطقي. إذا كانت القيمة الأولى صحيحة، يتم تعيين القيمة الثانية.

## ✓ مثال $\Downarrow\Downarrow$ منطقي والواجب

```
let x = 10;  
x &&= 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تم اصدارة في [ES2020](#) المشغل &&=

## || المشغل

بين قيمتين OR يتم استخدام عامل التعيين المنطقي. إذا كانت القيمة الأولى خاطئة، يتم تعيين القيمة الثانية.

## ✓ مثال $\Downarrow\Downarrow$ منطقي أو التنازل

```
let x = 10;  
x ||= 5;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تم اصدارة في [ES2020](#) المشغل ||=

## المشغل == ??

بين قيمتين Nullish يتم استخدام عامل تعيين الدمج. إذا كانت القيمة الأولى غير محددة أو فارغة، فسيتم تعيين القيمة الثانية.

مثال  $\Downarrow\Downarrow$  على مهمة الدمج الفارغة ✓

```
let x = 10;  
x ??= 5;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## أنواع البيانات العامة

جافاسكربت لديها 8 أنواع البيانات

1. نص
2. رقم
3. رقم كبير
4. منطقي
5. غير محدد
5. فارغ
7. رمز
8. كائن

نوع بيانات الكائن

يمكن أن يحتوي نوع بيانات الكائن على:

1. كائن
2. مصفوفة
3. تاريخ



## أمثلة على انشاء انواع من البيانات العامة

```
// Num:  
let length = 16;  
let weight = 7.5;  
  
// Strings:  
let color = "Yellow";  
let lastName = "tariq ibn Ziad";  
  
// Booleans  
let x = true;  
let y = false;  
  
// Object:  
const hAbiB = {firstName:"Habib", lastName:"Al Husini 🇸🇦🇸🇦🇸🇦"};  
  
// Array object:  
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];  
  
// Date object:  
const date = new Date("2022-03-25");
```

## ملحوظة

يمكن لمتغير جافاسكربت الاحتفاظ بأي نوع من البيانات. في المصفوفة وغير المصفوفة والتحول السريع

## مفهوم أنواع البيانات

في البرمجة، تعد أنواع البيانات مفهوماً مهماً. لتكون قادرًا على العمل على المتغيرات، من المهم أن تعرف شيئاً عن النوع: بدون أنواع البيانات، لا يمكن للكمبيوتر حل هذه المشكلة بأمان

```
let x = 16 + "Hamza";
```

هل من المنطقي إضافة "حمزا" إلى الستة عشر؟ هل سينتج خطأ أم سينتج نتيجة؟  
سوف تتعامل جافاسكربت مع ال ✓ مثال ❗ أعلاه على النحو التالي

```
let x = "16" + "Hamza";
```

## ملحوظة

• عند إضافة رقم ونص ، ستتعامل جافاسكربت مع الرقم كنص

✓ مثال ↓↓

```
let x = 16 + "Hamza";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

✓ مثال ↓↓

```
let x = "Hamza" + 16;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

تقوم جافاسكربت بتقييم التعبيرات من اليسار إلى اليمين. تسلسلات مختلفة يمكن أن تنتج نتائج مختلفة

```
let x = 16 + 4 + "Hamza";
```

نتيجة:

20Hamza

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

```
let x = "Hamza" + 16 + 4;
```

نتيجة:

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

"Hamza" في ال ✓ مثال ↓↓ الأول، تتعامل جافاسكربت مع الرقمين 16 و4 كأرقام، حتى تصل إلى .  
في ال ✓ مثال ↓↓ الثاني، بما أن المعامل الأول عبارة عن نص ، فسيتم التعامل مع جميع المعاملات كنصوص

## أنواع جافاسكربت الديناميكية

تحتوي جافاسكربت على أنواع ديناميكية، وهذا يعني أنه يمكن استخدام نفس المتغير للاحتفاظ بأنواع بيانات مختلفة أو : بمعنى اصح تستطيع تغيير النوع بمجرد اسناد قيمة النوع الجديد وهذا يعني ان جافاسكربت لغة سهلة جدا وذكية

✓ مثال ↓↓

```
let x; // Now x is undefined
x = 5; // Now x is a Number
x = "Habib"; // Now x is a String
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## نصوص جافاسكربت

"Abu Bakr Al-Siddiq" عبارة عن سلسلة نصية من الأحرف مثل  
:تتم كتابة النصوص مع علامات الاقتباس. يمكنك استخدام علامات الاقتباس المفردة أو المزدوجة

✓ مثال ↓↓

```
// Using double quotes:
let hAbiB41 = "Hamza ";

// Using single quotes:
let hAbiB42 = 'Hamza ';
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

: يمكنك استخدام علامات الاقتباس داخل النص ، طالما أنها لا تتطابق مع علامات الاقتباس المحيطة بالنص

✓ مثال ↓↓

```
// Single quote inside double quotes:
```

```
let Text1 = "It's alright";
```

```
// Single quotes inside double quotes:
```

```
let Text2 = "He is called 'habib'";
```

```
// Double quotes inside single quotes:
```

```
let Text3 = 'He is called "habib"';
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

. سوف تتعلم المزيد عن النصوص باذن الله تعالى في هذا الكتاب او في الجزء الثاني

## الأرقام في جافاسكريبت

يتم تخزين جميع أرقام جافاسكريبت كأرقام عشرية (نقطة عائمة)

:يمكن كتابة الأرقام مع أو بدون الكسور العشرية

✓ مثال ↓↓

```
// With decimals:
```

```
let x1 = 34.00;
```

```
// Without decimals:
```

```
let x2 = 34;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## الأسية في الجبر

يمكن كتابة الأعداد الكبيرة جدًا أو الصغيرة جدًا بالترميز العلمي (الأسية) او استخدام جميع ادوات الجبر لن نتعمق كثيرا في ادوات الجبر داخل الجافاسكريبت لانها تحتاج الى كتاب خاص بها ولكن شاهد هذا المثال البسيط في استخدام الاس العددى:

✓ مثال ↓↓

```
let y = 123e5; // 12300000  
let z = 123e-5; // 0.00123
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

انتظر كتاب خاص بادوات الجبر داخل الجافاسكربت وانشاء العمليات والمعادلات المعقدة باذن الله تعالى

## ملحوظة

تحتوي معظم لغات البرمجة على العديد من أنواع الأرقام

الأعداد الصحيحة (الأعداد الصحيحة)

الطويلة او الكبيرة (64 بت)، (32 بت) int، البايت (8 بت)، القصير (16 بت)

الأعداد الحقيقية (العشرية)

مزدوج (64 بت)، (32 بت)

مزدوج (نقطة عائمة 64 بت)

## ملحوظة

جافاسكربت هي دائماً نوع واحد: في الأرقام لن نستخدم غير لا تشغل نفسك بهذه الانقسامات والانواع الكثيرة فهي لها استخدامتها في الجبر والمعادلات الدقيقة فقط

## نوع البيانات BigInt

هذا النوع من البيانات ساشرحة من باب العلم فقط فهو نادر الاستخدام لانه مخصص للحسابات الفلكية الكبيرة داخل جافاسكربت وهو نوع بيانات جديد تم اصدرة في (2020) يمكن استخدامه لتخزين قيم عددية كبيرة جدًا للحسابات الفلكية وهذا يحتاج الى اجهزة كمبيوتر قوية جدا يطلق عليها وركستيشن بحيث لا يمكن تمثيلها برقم جافاسكربت عادي. يتم تخزين جميع أرقام جافاسكربت بتنسيق الفاصلة العشرية 64 بت.

✓ مثال ↓↓

```
let x = BigInt("123456789012345678901234567890");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## جافاسكربت و الإساليب المنطقية

`true` أو `false`: يمكن أن تحتوي القيم المنطقية على قيمتين صح او خطأ فقط.

✓ مثال ↓↓

```
let x = 5;  
let y = 5;  
let z = 6;  
(x == y) // Returns true  
(x == z) // Returns false
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

غالبًا ما تستخدم القيم المنطقية في الاختبارات الشرطية.

. سوف تتعلم المزيد عن الاختبار الشرطي باذن الله تعالى في هذا الكتاب او في الجزء الثانى

## المصفوفات في جافاسكربت

تتم كتابة مصفوقات جافاسكربت بين قوسين مربعين.

يتم فصل عناصر المصفوفة بفواصل

:تحتوي على ثلاثة عناصر (أسماء) **HaBiB**، يعلن الاكواد التالية بنشا مصفوفة تسمى

✓ مثال ↓↓

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

تعتمد فهرس المصفوقات على الصفر، مما يعني أن العنصر الأول هو [0]، والثاني هو [1]، وهكذا

. سوف تتعلم المزيد عن المصفوقات باذن الله تعالى في هذا الكتاب او في الجزء الثاني

## الكائنات في جافاسكربت

👉 تتم كتابة كائنات جافاسكربت باستخدام الأقواس المتعرجة

.تتم كتابة خصائص الكائن على هيئة أزواج الاسم: القيمة، مفصولة بفواصل

✓ مثال ↓↓

```
const hAbiB = {firstName:"Habib", lastName:"Al Husini 🇸🇦🇸🇦🇸🇦", age:70, eyeColor:"blue👁️👁️👁️"};
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

.الكائن (الشخص) في ال ✓ مثال ↓↓ أعلاه له 4 خصائص: الاسم الأول، الاسم الأخير، العمر، ولون العين

. سوف تتعلم المزيد عن الكائنات باذن الله تعالى في هذا الكتاب او في الجزء الثاني

## كيف معرفة الأنواع للبيانات

عامل تشغيل جافاسكربت للعثور على نوع متغير جافاسكربت **typeof** يمكنك استخدام

:التشغيل بإرجاع نوع المتغير أو التعبير **typeof** يقوم عامل

✓ مثال ↓↓

```
typeof "" // Returns "string"
typeof "Habib" // Returns "string"
typeof "Abu Bakr Al-Siddiq" // Returns "string"
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

✓ مثال ↓↓

```
typeof 0 // Returns "number"
typeof 314 // Returns "number"
typeof 3.14 // Returns "number"
typeof (3) // Returns "number"
typeof (3 + 4) // Returns "number"
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

. باذن الله تعالى في هذا الكتاب او في الجزء الثاني **typeof** سوف تتعلم المزيد من الامثلة عن

## الإعلان عن نوع غير معرف وليس له قيمة

. النوع هو **undefined** في جافاسكربت، المتغير بدون قيمة لة

✓ مثال ↓↓

```
let $_Habib; // Value is undefined, type is undefined
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

**undefined** سيكون النوع أيضًا. **undefined** يمكن إفراغ أي متغير عن طريق ضبط القيمة على



✓ مثال ↓↓

```
$_Habib= undefined; // Value is undefined, type is undefined
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## القيم الفارغة

**undefined** القيمة الفارغة ليس لها علاقة بـ  
تحتوي النص الفارغة على قيمة قانونية ونوع

✓ مثال ↓↓

```
let $_Habib= ""; // The value is "", the typeof is "string"
```

## الوظائف في جافاسكربت

وظيفة جافاسكربت عبارة عن كتلة من التعليمات البرمجية مصممة لأداء مهمة معينة  
يتم تنفيذ وظيفة جافاسكربت عندما يستدعيها "شيء ما" (يستدعيها)

✓ مثال ↓↓

```
// Function to compute the product of p1 and p2
function Husini(p1, p2) {
  return p1 * p2;
}
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## كيف بناء جملة الوظيفة

المحجوزة، متبوعة بالاسم ، متبوعاً بالأقواس **function** يتم تعريف دالة جافاسكربت باستخدام الكلمة  
يمكن أن تحتوي أسماء الوظائف على أحرف وأرقام وشرطات سفلية وعلامات الدولار (نفس قواعد المتغيرات)

قد تتضمن الأقواس أسماء الرموز مفصولة بفواصل  
( parameter1,parameter2,... )

{ : يتم وضع الكود الذي سيتم تنفيذه بواسطة الوظيفة داخل قوسين متعرجين

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

يتم إدراج معلمات الوظيفة داخل الأقواس ( ) في تعريف الوظيفة

وسيطات الدالة هي القيم التي تتلقاها الدالة عند استدعائها

داخل الدالة، تعمل الوسيطات (الرموز) كمتغيرات محلية

## كيف استدعاء الوظيفة

:سيتم تنفيذ التعليمات البرمجية الموجودة داخل الوظيفة عندما يقوم " ربط حدث ما " باستدعاء الوظيفة

- عند وقوع حدث (عندما ينقر المستخدم على زر)
- عندما يتم استدعاؤه من كود جافاسكربت
- تلقائياً (الاستدعاء الذاتي) داخل الاقواس وهذا يعنى تنفيذ مباشر

. سوف تتعلم الكثير عن استدعاء الوظائف باذن الله تعالى في هذا الكتاب او فى الجزء الثانى

## التحكم فى الإرجاع من داخل الوظيفة

.عبارة، ستتوقف الوظيفة عن التنفيذ **return** عندما تصل جافاسكربت إلى

.إذا تم استدعاء الوظيفة من عبارة، فسوف تعود جافاسكربت لتنفيذ التعليمات البرمجية بعد عبارة الاستدعاء

:" غالبًا ما تحسب الوظائف قيمة الإرجاع . يتم "إرجاع" القيمة المرجعة إلى "المتصل

مثال ↓↓ ✓

:حساب حاصل ضرب رقمين وإرجاع النتيجة

```
let x = Husini(4, 3); // Function is called, return value will end up in x
```

```
function Husini(a, b) {  
  return a * b; // Function returns the product of a and b  
}
```

ستكون x النتيجة في

12

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## لماذا نستخدم الوظائف؟

يمكنك إعادة استخدام الكود: حدد الكود مرة واحدة، واستخدمه عدة مرات. وبالتالي هي مهمة جدا لتوفير الوقت والمجهود. يمكنك استخدام نفس الكود عدة مرات مع وسائط مختلفة للحصول على نتائج مختلفة.

✓ مثال ↓↓

تحويل فهرنهايت إلى مئوية:

```
function toCelsius(fahrenheit) {  
  return (5/9) * (fahrenheit-32);  
}  
document.getElementById("Habib").innerHTML = toCelsius(77);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## استخدام الوظيفة لإرجاع ناتج العملية

ويشير إلى نتيجة الوظيفة `toCelsius()`، يشير إلى كائن الوظيفة `toCelsius`، باستخدام ال `toCelsius` مثال ↓↓ أعلاه سيؤدي الوصول إلى وظيفة بدون إلى إرجاع كائن الوظيفة بدلاً من نتيجة الوظيفة.

✓ مثال ↓↓

```
function toCelsius(fahrenheit) {  
  return (5/9) * (fahrenheit-32);  
}  
document.getElementById("Habib").innerHTML = toCelsius;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف استخدام الوظائف كقيم متغيرة

يمكن استخدام الوظائف بنفس الطريقة التي تستخدم بها المتغيرات، في كافة أنواع الصيغ والواجبات والحسابات.

✓ مثال ↓↓

بدلاً من استخدام متغير لتخزين القيمة المرجعة للدالة:

```
let x = toCelsius(77);  
let text = "The temperature is " + x + " Celsius";
```

يمكنك استخدام الدالة مباشرة كقيمة متغيرة:

```
let text = "The temperature is " + toCelsius(77) + " Celsius";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

. سوف تتعلم الكثير عن الوظائف باذن الله تعالى في هذا الكتاب او في الجزء الثانى

## نطاق المتغيرات المحلية

المتغيرات المعلنة داخل وظيفة جافاسكربت، تصبح محلية للوظيفة.

لا يمكن الوصول إلى المتغيرات المحلية إلا من داخل الوظيفة.

✓ مثال ↓↓

```
// code here can NOT use hAbiB4
```

```
function Husini() {  
  let hAbiB4 = "Hamza";  
  // code here CAN use hAbiB4  
}
```

```
// code here can NOT use hAbiB4
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

وبما أنه يتم التعرف على المتغيرات المحلية فقط داخل وظائفها، فيمكن استخدام المتغيرات التي تحمل الاسم نفسه في وظائف مختلفة.

يتم إنشاء المتغيرات المحلية عند بدء تشغيل الدالة، ويتم حذفها عند اكتمال الدالة.

## كائنات جافاسكريبت

لقد تعلمت بالفعل أن متغيرات جافاسكريبت عبارة عن حاويات لقيم البيانات.

**\$\_Habib**: لمتغير اسمه (Mohamed) يعين هذا الكود قيمة بسيطة

```
let $_Habib = "Mohamed";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

الكائنات هي متغيرات أيضا، لكن الكائنات يمكن أن تحتوي على العديد من القيم.

**\$\_Habib**: لمتغير اسمه (Mohamed, 500, White) يعين هذا الكود العديد من القيم

```
const $_Habib = {use:"Mohamed", age:"20", color:"white"};
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تتم كتابة القيم على هيئة أزواج الاسم: القيمة (الاسم والقيمة مفصولة بنقطتين)

**const** . من الممارسات الشائعة الإعلان عن الكائنات باستخدام الكلمة المحجوزة

**JS Const**: مع الكائنات في الفصل **const** تعرف على المزيد حول استخدام

## تعريف الكائن

يمكنك تعريف (وإنشاء) كائن جافاسكربت بكائن حرفي

✓ مثال ↓↓

```
const hAbiB = {firstName:"Habib", lastName:"Al Husini 🇸🇩🇸🇩🇸🇩", age:50, eyeColor:"blue"};
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

المسافات وفواصل الأسطر ليست مهمة. يمكن أن يمتد تعريف الكائن على عدة أسطر

✓ مثال ↓↓

```
const hAbiB = {
  firstName: "Habib",
  lastName: "Al Husini 🇸🇩🇸🇩🇸🇩",
  age: 50,
  eyeColor: "blue"
};
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## خصائص الكائنات

الخصائص هي عبارة مجموعة من القيم تصف الكائن

## الوصول إلى خصائص الكائن

يمكنك الوصول إلى خصائص الكائن بطريقتين

```
objectName.propertyName
```

أو

```
objectName["propertyName"]
```

## مثال 1 ↓↓ ✓

```
hAbiB.lastName;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## مثال 2 ↓↓ ✓

```
hAbiB["lastName"];
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

كائنات جافاسكريبت عبارة عن حاويات لقيم مسماة تسمى الخصائص

## طرق الكائن

- يمكن أن تحتوي الكائنات أيضًا على طرق.
- الأساليب هي الإجراءات التي يمكن تنفيذها على الكائنات.
- يتم تخزين الأساليب في الخصائص كتعريفات الوظيفة.
- الطريقة هي وظيفة مخزنة كخاصية.

## مثال ↓↓ ✓

```
const hAbiB = {  
  firstName: "Habib",  
  lastName : "Al Husini 😊😊😊",  
  id : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

يشير إلى كائن الشخص **this** في ال ✓ مثال ↓↓ أعلاه

يعني خاصية الاسم الأول لهذا IE this.firstName

IE `this.firstName` يعني خاصية الاسم الأول للشخص

## كيف الإشارة الى الكائن بدون ذكر اسمة تشير الكلمة المحجوزة إلى كائن `this` .

استدعائه (استخدامه أو استدعائه) `this` يعتمد الكائن على كيفية  
المحجوزة إلى كائنات مختلفة اعتمادًا على كيفية استخدامها `this` تشير الكلمة

### ملحوظة

`this` ليس متغيراً. إنها كلمة محجوزة. لا يمكنك تغيير قيمة `this`.

### أنظر أيضا:

[اقراء قسم الكلاسات في كتاب الكافي في php](#)

## كيف استخدام كلمة `this`

يشير إلى "مالك" الوظيفة `this`، في تعريف الوظيفة  
الوظيفة `fullName` هو كائن الشخص الذي "يمتلك `this`، في ال `✓` مثال `↓ ↓` أعلاه  
. خاصية هذا الكائن `firstName` يعني `this.firstName`، وبعبارة أخرى  
. في كتاب المرجع الكامل لجافاسكربت `this` تعرف على المزيد حول الكلمة

## كيف الوصول الى اعضاء الكائن

يمكنك الوصول إلى أسلوب كائن باستخدام بناء الجملة التالي:

```
objectName.methodName()
```



✓ مثال ↓↓

```
name = hAbiB.fullName();
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة

: إذا قمت بالوصول إلى طريقة بدون الأقواس (), فسوف تُرجع تعريف الوظيفة

✓ مثال ↓↓

```
name = hAbiB.fullName;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ! تستخدم هذه الطريقة

! لا تعلن عن النصوص والأرقام والقيم المنطقية ككائنات باستخدام الكلمة نيو

يتم إنشاء المتغير ككائن، "new" عندما يتم الإعلان عن متغير جافاسكربت باستخدام الكلمة المحجوزة

```
x = new String(); // Declares x as a String object  
y = new Number(); // Declares y as a Number object  
z = new Boolean(); // Declares z as a Boolean object
```

الأشياء. إنها تعقد التعليمات البرمجية وتبطئ سرعة التنفيذ Boolean و Number String تجنب

. سوف تتعلم المزيد عن الكائنات باذن الله تعالى في هذا الكتاب او في الجزء الثانى

## أحداث جافاسكربت

HTML هي "أشياء" تحدث لعناصر HTML أحداث

يمكن لـ جافاسكربت "التفاعل" مع هذه الأحداث، HTML عند استخدام جافاسكربت في صفحات

## HTML أحداث

يمكن أن يكون حدث أي شيئًا يفعله المتصفح، أو أي شيئًا يفعله المستخدم.

**HTML:** فيما يلي بعض الأمثلة على أحداث

- HTML تم الانتهاء من تحميل صفحة ويب
- HTML تم تغيير حقل إدخال
- HTML تم النقر على زر

في كثير من الأحيان، عندما تحدث الأحداث، قد ترغب في القيام بشيء ما

تتيح لك جافاسكربت تنفيذ التعليمات البرمجية عند اكتشاف الأحداث

**HTML:** بإضافة سمات معالج الأحداث، مع كود جافاسكربت، إلى عناصر HTML يسمح

مع علامات الاقتباس المفردة

```
<element event='some javascript'>
```

مع علامات الاقتباس المزدوجة

```
<element event="some javascript">
```

✓ مثال ↓↓

```
<button onclick="document.getElementById('Habib').innerHTML = Date()">The time is?</button>
```

عند الضغط على الزر سيعرض الوقت الحالي

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

"Habib" = في ال ✓ مثال ↓↓ أعلاه، يقوم كود جافاسكربت بتغيير محتوى العنصر بالمعرف

(**this.innerHTML**) باستخدام) في ال ✓ مثال ↓↓ التالي، يغير الكود محتوى العنصر الخاص به

✓ مثال ↓↓

```
<button onclick="this.innerHTML = Date()">The time is?</button>
```

غالبًا ما يتكون كود جافاسكربت من عدة أسطر. من الأكثر شيوعًا رؤية سمات الحدث التي تستدعي الوظائف

مثال ↓↓ ✓

```
<button onclick="displayDate()">The time is?</button>
```

## أحداث الشائعة لـ HTML

الشائعة HTML فيما يلي قائمة ببعض أحداث

Event	وصف
onchange	HTML تم تغيير عنصر
onclick	HTML يقوم المستخدم بالنقر فوق عنصر
onmouseover	HTML يقوم المستخدم بتحريك الماوس فوق عنصر
onmouseout	HTML يقوم المستخدم بتحريك الماوس بعيدًا عن عنصر
onkeydown	يقوم المستخدم بالضغط على مفتاح لوحة المفاتيح
onload	لقد انتهى المتصفح من تحميل الصفحة

## معالجات أحداث جافاسكربت

يمكن استخدام معالجات الأحداث للتعامل مع إدخلات المستخدم وإجراءات المستخدم وإجراءات المتصفح والتحقق منها

- الأشياء التي يجب القيام بها في كل مرة يتم فيها تحميل الصفحة
- الأشياء التي يجب القيام بها عند إغلاق الصفحة
- الإجراءات التي يجب تنفيذها عندما ينقر المستخدم على زر
- المحتوى الذي يجب التحقق منه عندما يقوم المستخدم بإدخال البيانات
- ... و أكثر

يمكن استخدام العديد من الطرق المختلفة للسماح لـ جافاسكربت بالعمل مع الأحداث

- تنفيذ كود جافاسكربت مباشرة HTML يمكن سمات حدث
- استدعاء وظائف جافاسكربت HTML يمكن سمات حدث

- HTML يمكنك تعيين وظائف معالج الأحداث لعناصر
- يمكنك منع إرسال الأحداث أو معالجتها
- ... و اكثر

. سوف تتعلم الكثير عن الأحداث ومعالجات الأحداث في كتاب الكافي في HTML

## التعامل مع النصوص

نصوص جافاسكربت مخصصة لتخزين النص ومعالجته

نص جافاسكربت عبارة عن صفر أو أكثر من الأحرف المكتوبة داخل علامات الاقتباس

✓ مثال ↓↓

```
let text = "Abu Bakr Al-Siddiq";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

:يمكنك استخدام علامات الاقتباس المفردة أو المزدوجة

✓ مثال ↓↓

```
let hAbiB41 = "Hamza "; // Double quotes
let hAbiB42 = 'Hamza '; // Single quotes
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

: يمكنك استخدام علامات الاقتباس داخل النص ، طالما أنها لا تتطابق مع علامات الاقتباس المحيطة بالنص

✓ مثال ↓↓

```
let Text1 = "It's alright";
let Text2 = "He is called 'habib'";
let Text3 = 'He is called "habib"';
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيف معرفة عدد حروف النصوص

**length:** للعثور على طول نص ، استخدم الخاصية المضمنة

✓ مثال ↓↓

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = text.length;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف استثناء بعض الحروف

نظرًا لأنه يجب كتابة النصوص بين علامتي اقتباس، فسوف تسيء جافاسكريبت فهم هذه النص

```
let text = "We are the so-called "Vikings" from the north.";
```

سيتم قطع النص إلى نهاية العلامة

. الحل لتجنب هذه المشكلة هو استخدام حرف الاستثناء للشرطة المائلة العكسية

: يقوم حرف الاستثناء الخاص بالشرطة المائلة العكسية ( \ ) بتحويل الأحرف الخاصة إلى أحرف نص

Code	Result	وصف
'	'	اقتباس واحد
"	"	اقتباس مزدوج
\	\	شرطة مائلة عكسية

: يُدرج التسلسل \ " علامة اقتباس مزدوجة في نص

✓ مثال ↓↓

```
let text = "We are the so-called \"Vikings\" from the north.";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

: يقوم التسلسل \ بإدراج اقتباس واحد في نص

✓ مثال ↓↓

```
let text = 'It\'s alright.';
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

: يُدرج التسلسل \\ شرطة مائلة عكسية في نص

✓ مثال ↓↓

```
let text = "The character \\ is called backslash.";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

: ستة تسلسلات هروب أخرى صالحة في جافاسكربت

Code	نتيجة
\b	مسافة للخلف
\f	نموذج اضافة
\n	خط جديد
\r	إرجاع
\t	جدول أفقي
\v	جدول عمودي

تم تصميم أحرف الاستثناء الستة المذكورة أعلاه في الأصل للتحكم في الآلات الكاتبة وأجهزة المبرقة وأجهزة الفاكس. أنها لا HTML معنى لها في

## كيف تحديد طول السطر

للحصول على أفضل سهولة للقراءة، غالبًا ما يرغب المبرمجون في تجنب سطور التعليمات البرمجية التي يزيد طولها عن 80 حرفًا.

:إذا كانت عبارة جافاسكربت لا تتناسب مع سطر واحد، فإن أفضل مكان لفصلها هو بعد عامل التشغيل

✓ مثال ↓↓

```
document.getElementById("Habib").innerHTML =  
"Abu Habib Al-Hosini";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

:يمكنك أيضًا تقسيم سطر التعليمات البرمجية داخل نصوص باستخدام شرطة مائلة عكسية واحدة

✓ مثال ↓↓

```
document.getElementById("Habib").innerHTML = "Easy-to \\  
HABIB!";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

الطريقة \ ليست الطريقة المفضلة، ربما لا تحظى بدعم عام  
بعض المتصفحات لا تسمح بمسافات خلف \ الحرف

: الطريقة الأكثر أمانًا لتقسيم النص هي استخدام إضافة النص

✓ مثال ↓↓

```
document.getElementById("Habib").innerHTML = "Easy-to " +  
"HABIB!";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

:لا يمكنك تقسيم سطر التعليقات البرمجية بشرطة مائلة عكسية

✓ مثال ↓↓

```
document.getElementById("Habib").innerHTML = \  
"Abu Habib Al-Hosini!";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## طريقة استخدام النصوص ككائنات

:عادةً ما تكون نصوص جافاسكربت عبارة عن قيم بدائية، يتم إنشاؤها من القيم الحرفية

```
let x = "Habib";
```

**new**: ولكن يمكن أيضًا تعريف النصوص ككائنات باستخدام الكلمة المحجوزة

```
let y = new String("Habib");
```

✓ مثال ↓↓

```
let x = "Habib";  
let y = new String("Habib");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

لا تقم بإنشاء كائنات نصوص

المحجوزة على تعقيد التعليمات البرمجية وإبطاء سرعة التنفيذ **new** تعمل الكلمة

:يمكن أن تنتج كائنات النص نتائج غير متوقعة

: متساويان لا **x**، عند استخدام **==** المعامل

```
let x = "Habib";  
let y = new String("Habib");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

: غير متساويين لا **x**، عند استخدام **===** المعامل

```
let x = "Habib";  
let y = new String("Habib");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

، **(x==y)** و **(x===y)** لاحظ الفرق بين

صحيحة أو خاطئة؟ **(x == y)**

```
let x = new String("Habib");  
let y = new String("Habib");
```



صحيحة أو خاطئة؟ (x === y)

```
let x = new String("Habib");  
let y = new String("Habib");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

. false دائماً ما تؤدي المقارنة بين كائنين في جافاسكربت إلى إرجاع

## دوال النصوص

### ملحوظة

إذا اردة الحصول على شرح كامل لدوال جافاسكربت اقرا كتاب المرجع الكامل في الجافاسكربت

. يحتوي المرجع على أوصاف وأمثلة لجميع الخصائص والدوال والأساليب

## شرح بعض دوال النصوص في هذا الكتاب

String length

String slice()

String substring()

String substr()

String replace()

String replaceAll()

String toUpperCase()

String toLowerCase()

String concat()

String trim()

String trimStart()

String trimEnd()

String padStart()

String padEnd()

String charAt()

String charCodeAt()

String split()

يتم تناول طرق البحث عن النص في الفصل التالي

## الدالة length

: بلرجاع طول النص **length** تقوم الخاصية

✓ مثال ↓↓

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = text.length;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف استخراج أجزاء من النص

: هناك ثلاث طرق لاستخراج جزء من النص

- **slice(start, end)**
- **substring(start, end)**
- **substr(start, length)**

## طرق التقطيع الدالة الأولى

يستخرج جزءاً من النص ويعيد الجزء المستخرج في نص جديدة (**slice()**)

تأخذ الطريقة معلمتين: موضع البداية، وموضع النهاية (النهاية غير متضمنة)

✓ مثال ↓↓

:قم بقطع جزء من النص من الموضع 7 إلى الموضع 13

```
let text = "Abu Habib, Al-Husini, Hamza";  
let part = text.slice(7, 13);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ملحوظة

جافاسكربت تحسب المواضع من الصفر

المركز الأول هو 0

المركز الثاني هو 1

### أمثلة

إذا قمت بحذف المعلمة الثانية، فسوف تقوم الطريقة بتقسيم بقية النص

```
let text = "Abu Habib, Al-Husini, Hamza";  
let part = text.slice(7);
```

كتاب الكافي في جافاسكربت الجزء الأول ابو حبيب الحسيني

إذا كانت المعلمة سالبة، فسيتم حساب الموضع من نهاية النص

```
let text = "Abu Habib, Al-Husini, Hamza";  
let part = text.slice(-12);
```

كتاب الكافي في جافاسكربت الجزء الأول ابو حبيب الحسيني

يقطع هذا ال ✓ مثال ↓↓ جزءاً من النص من الموضع -12 إلى الموضع -6

```
let text = "Abu Habib, Al-Husini, Hamza";  
let part = text.slice(-12, -6);
```

كتاب الكافي في جافاسكربت الجزء الأول ابو حبيب الحسيني

## تابع طرق تقطيع النصوص

**substring()** مشابه ل **slice()**.

**substring()** الفرق هو أن قيم البداية والنهاية الأقل من 0 يتم التعامل معها على أنها 0 في

✓ مثال ↓↓

```
let str = "Abu Habib, Al-Husini, Hamza";  
let part = str.substring(7, 13);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

. سيتم قطع بقية النص **substring()**، إذا قمت بحذف المعلمة الثانية

## دالة Substr

**substr()** مشابه ل **slice()**.

الفرق هو أن المعلمة الثانية تحدد طول الجزء المستخرج.

✓ مثال ↓↓

```
let str = "Abu Habib, Al-Husini, Hamza";  
let part = str.substr(7, 6);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

. سيتم قطع بقية النص **substr()**، إذا قمت بحذف المعلمة الثانية

✓ مثال ↓↓

```
let str = "Abu Habib, Al-Husini, Hamza";  
let part = str.substr(7);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

. إذا كانت المعلمة الأولى سلبية، يتم حساب الموضع من نهاية النص

✓ مثال ↓↓

```
let str = "Abu Habib, Al-Husini, Hamza";  
let part = str.substr(-4);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## كيف البحث والاستبدال في النصوص

: قيمة محددة بقيمة أخرى في نص `replace()` تستبدل الطريقة

✓ مثال ↓↓

```
let text = "Please visit Hamza!";  
let newText = text.replace("Hamza", "Abu Habib alhosiny");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## ملحوظة 1

. لا تغير طريقة النص التي يتم استدعاؤها عليها `replace()`

. يرجع نص جديدة `replace()` تقوم الطريقة

تستبدل الطريقة المطابقة الأولى فقط `replace()`

. انظر الأمثلة أدناه `/g` إذا كنت تريد استبدال كافة التبادلات، فاستخدم تعبيرًا عاديًا مع مجموعة العلامات

: تستبدل الطريقة المطابقة الأولى فقط `replace()`، افتراضيًا

✓ مثال ↓↓

```
let text = "Please visit Hamza and Omar!";  
let newText = text.replace("Hamza", "Abu Habib alhosiny");
```

سيتم استبدال حمزا الاولى فقط

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

(بالأحرف الكبيرة) Hamza الأسلوب حساس لحالة الأحرف. لن تنجح كتابة **replace()** بشكل افتراضي

✓ مثال ↓↓

```
let text = "Please visit hamza!";  
let newText = text.replace("Hamza", "Abu Habib alhosiny");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

(غير حساسة) **/i** لاستبدال الحالة غير الحساسة لحالة الأحرف، استخدم تعبيرًا عاديًا بعلامة

✓ مثال ↓↓

```
let text = "Please visit Hamza!";  
let newText = text.replace(/Hamza/i, "Abu Habib alhosiny");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة 2

تتم كتابة التعبيرات العادية دون علامات الاقتباس

(مطابقة عامة) **/g** لاستبدال جميع التطابقات، استخدم تعبيرًا عاديًا بعلامة

✓ مثال ↓↓

```
let text = "Please visit Hamza and Hamza!";  
let newText = text.replace(/Hamza/g, "Abu Habib alhosiny");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة 3

سوف تتعلم الكثير عن دوال النصوص والتحكم الكامل في كتاب المرجع الكامل لدوال جافاسكربت

## دالة ()replaceAll

**replaceAll()** في عام 2021، قدمت جافاسكربت طريقة لاستبدال جميع التطابقات في النص

مثال ↓↓ ✓

```
text = text.replaceAll("Habib","Mahmoud");  
text = text.replaceAll("Habib","Mahmoud");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

تحديد تعبير عادي بدلاً من نص ليتم استبدالها **replaceAll()** تتيح لك الطريقة  
وإلا فسيتم طرح خطأ، **(g)** إذا كانت المعلمة تعبيراً عادياً، فيجب تعيين العلامة العامة

مثال ↓↓ ✓

```
text = text.replaceAll(/Habib/g,"Mahmoud");  
text = text.replaceAll(/Habib/g,"Mahmoud");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة 4

لا يعمل في إنترنت إكسبلورر **replaceAll()**

## التحويل إلى الأحرف الكبيرة والصغيرة

**toUpperCase()** يتم تحويل النص إلى أحرف كبيرة باستخدام

**toLowerCase()** يتم تحويل النص إلى أحرف صغيرة باستخدام

## دالة ( )toUpperCase

✓ مثال ↓↓

```
let text1 = "Abu Habib Al Husini * _*!";  
let text2 = text1.toUpperCase(); // ABU HABIB AL HUSINI * _*
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## دالة ( )toLowerCase

✓ مثال ↓↓

```
let text1 = "ABU HABIB AL HUSINI * _*!"; // String  
let text2 = text1.toLowerCase(); // text2 is text1 converted to lower
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## الدالة concat

دمج النصوص والمتغيرات والبيانات نصين أو أكثر `concat()`

✓ مثال ↓↓

```
let text1 = "Easy-to";  
let text2 = "Arabic";  
let text3 = text1.concat(" ", text2, " ", " ", text1);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

هذان الخطان يفعلان نفس الشيء . `plus` يمكن استخدام الطريقة بدلاً من عامل التشغيل `concat()`

✓ مثال ↓↓

```
text = "Easy-to" + " " + "Arabic!";  
text = "Easy-to".concat(" ", "Arabic!");
```



## ملحوظة

تقوم كافة أساليب النص بإرجاع نص جديدة. لا يقومون بتعديل النص الأصلية.

قال رسميًا

النصوص غير قابلة للتغيير: لا يمكن تغيير النصوص، بل استبدالها فقط

## دالة Trim()

:بإزالة المسافة الفارغة من كلا جانبي النص **trim()** تقوم الطريقة

✓ مثال ↓↓

```
let text1 = " Abu Habib Al Husini *_*! ";  
let text2 = text1.trim();
```

المسافة الفارغة

كتاب الكافي في جافاسكربت الجزء الأول ابو حبيب الحسيني

## الدالة TrimStart ()

.ولكنها تزيل المسافات البيضاء فقط من بداية النص **trim()** ، تعمل الطريقة مثل **trimStart()**

✓ مثال ↓↓

```
let text1 = " Abu Habib Al Husini *_*! ";  
let text2 = text1.trimStart();
```

كتاب الكافي في جافاسكربت الجزء الأول ابو حبيب الحسيني

مدعومة في جميع المتصفحات منذ يناير 2020 **trimStart()** نص جافاسكربت

## دالة () TrimEnd

ولكنها تزيل المسافات البيضاء فقط من نهاية النص `trim()` ، تعمل الطريقة مثل `trimEnd()`

✓ مثال ↓↓

```
let text1 = " Abu Habib Al Husini * _! ";  
let text2 = text1.trimEnd();
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

مدعومة في جميع المتصفحات منذ يناير 2020 `trimEnd()` نص جافاسكربت

## دوال اخرى للإضافة

الربط في بداية النص وفي نهايتها `padEnd()` ولدعم `padStart()` : طريقتين للنص ECMAScript 2017 أضاف

## دالة () PadStart

على ربط نص بنص أخرى `padStart()` تعمل الطريقة

✓ مثال ↓↓

```
let text = "5";  
let padded = text.padStart(4,"x");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

✓ مثال ↓↓

```
let text = "5";  
let padded = text.padStart(4,"0");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة

لإضافة رقم، قم بتحويل الرقم إلى نص أولاً

انظر ال ✓ مثال ↓↓ أدناه

✓ مثال ↓↓

```
let numb = 5;  
let text = numb.toString();  
let padded = text.padStart(4,"0");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## دعم المتصفحات

padStart() هي إحدى ميزات ECMAScript 2017.

وهو مدعوم في جميع المتصفحات الحديثة

## الدالة ( ) PadEnd

على ربط نص بنص أخرى padEnd() تعمل الطريقة

✓ مثال ↓↓

```
let text = "5";  
let padded = text.padEnd(4,"x");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

✓ مثال ↓↓

```
let text = "5";  
let padded = text.padEnd(4,"0");
```

## ملحوظة

مثل الدالة السابقة تمام لإضافة رقم، قم بتحويل الرقم إلى نص أولاً

انظر ال ✓ مثال ↓↓ أدناه

✓ مثال ↓↓

```
let numb = 5;  
let text = numb.toString();  
let padded = text.padEnd(4,"0");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## دعم المتصفحات

**padEnd()** تمت الاضافة في ECMAScript 2017.

نوهو مدعوم في جميع المتصفحات الحديثة

Internet Explorer غير مدعوم في **padEnd()**.

## كيف استخراج أحرف معينة من النص

: هناك ثلاث طرق لاستخراج أحرف النص

- **charAt(position)**
- **charCodeAt(position)**
- الوصول إلى الخصائص [ ]

## الدالة () charAt

: بإرجاع الحرف في فهرس (موضع) محدد في نص **charAt()** تقوم الطريقة

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini * _*";  
let char = text.charAt(0);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## الدالة charCodeAt()

: بإرجاع الكود الموحد للحرف في فهرس محدد في نص charCodeAt() تقوم الطريقة  
(عدد صحيح بين 0 و65535) UTF-16 تقوم الطريقة بإرجاع رمز

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini * _*";  
let char = text.charCodeAt(0);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## جلب الحرف بالفهرس

:بالوصول إلى الخاصية [ ] على النصوص ECMAScript 5 (2009) يسمح

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini * _*";  
let char = text[0];
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة

:قد يكون الوصول إلى الخصائص غير متوقع إلى حد ما

- يجعل النصوص تبدو مثل المصفوفات (لكنها ليست كذلك)

- نص فارغة (`charAt()`) إذا لم يتم العثور على أي حرف، فإن `[ ]` يُرجع غير محدد، بينما يُرجع
- لا يعطي أي خطأ (لكنه لا يعمل!) `str[0] = "A"`، يتم قراءته فقط

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini * _ *";
text[0] = "A"; // Gives no error, but Al Husini 😊😊😊s not work
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## تحويل نص إلى مصفوفة

إذا كنت تريد العمل مع نص كمصفوفة، فيمكنك تحويلها إلى مصفوفة بسهولة وبطرق عديدة.

## الطريقة العادية للتحويل

الطريقة (`split()`) يمكن تحويل النص إلى مصفوفة باستخدام

✓ مثال ↓↓

```
text.split(",") // Split on commas
text.split(" ") // Split on spaces
text.split("|") // Split on pipe
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

إذا تم حذف الفاصل، فإن المصفوفة التي تم إرجاعها ستحتوي على النص بأكملها في الفهرس `[0]`.  
إذا كان الفاصل هو `""`، فستكون المصفوفة التي تم إرجاعها عبارة عن مصفوفة من الأحرف المفردة

✓ مثال ↓↓

```
text.split("")
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كتاب المرجع الكامل في الجافاسكريبت

للتعمق في جميع دوال المصفوفة والنصوص والتحكم الكامل لكل الدوال بلا استثناء

يحتوي المرجع على أوصاف وأمثلة لجميع خصائص النص وأساليبها

### كيف جلب الفهرس للحرف

: فهرس (موضع) التواجد الأول لنص في نص `indexOf()` تُرجع الطريقة

مثال ↓↓ ✓

```
let str = "Abu Habib Al Husini *_*";  
str.indexOf("Habib");
```

كتاب الكافي في جافاسكريبت الجزء الأول ابو حبيب الحسيني

### ملحوظة

نعيد للتذكير جافاسكريبت تحسب المواضع من الصفر

...، هو الموضع الأول في النص ، 1 هو الثاني، 2 هو الثالث 0

## الدالة lastIndexOf ()

: بإرجاع فهرس آخر تواجد لنص محدد في نص **lastIndexOf()** تقوم الطريقة

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini Habib *_*";  
text.lastIndexOf("Habib");
```

سيتم جلب حبيب الاخرة فقط

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

نوارجاع -1 إذا لم يتم العثور على النص **lastIndexOf()**، **indexOf()** كلاهما

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini *_* Habib Habib ";  
text.lastIndexOf("Habib");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

:تقبل كلتا الطريقتين معلمة ثانية كنقطة بداية للبحث

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini *_* Habib ";  
text.indexOf("Habib", 15);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

بشكل عكسي (من النهاية إلى البداية)، أي: إذا كان المعامل الثاني هو 15، يبدأ البحث من **lastIndexOf()** تبحث الطرق .الموضع 15، ويبحث حتى بداية النص

✓ مثال ↓↓

```
let text = "Abu Habib Al Husini *_*";  
text.lastIndexOf("Habib", 15);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني



## استخدام دالة البحث

عن نص (أو تعبير عادي) وترجع موضع المطابقة (`search()`) تبحث الطريقة

### أمثلة

```
let str = "Abu Habib Al Husini *_*";  
str.search("Habib");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

```
let str = "Abu Habib Al Husini *_*";  
str.search(/Habib/);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ملحوظة

هل الدالتين متساويتين؟، `search()` و `indexOf()`.

الطريقتين ليستا متساويتين. كما تظن: لن اخبرك بالاختلاف جربة بنفسك وستعرفه من النتيجة

## كيف البحث الاحترافي في النصوص

بإرجاع مصفوفة تحتوي على نتائج مطابقة نص مع نص (أو تعبير عادي) `match()` تقوم الطريقة

### أمثلة

:"إجراء بحث عن "عين

```
let text = "Abu Habib Al Husini *_*_*_*_*_*";  
text.match("Habib");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

"إجراء بحث عن "حبيب

```
let text = "Abu Habib Al Husini *_ *_ *_ *_*";  
text.match(/Habib/);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

"إجراء بحث عام عن "حبيب لكل التطابقات

```
let text = "Abu Habib Al Husini *_ * Habib* *_ Habib* *_*";  
text.match(/Habib/g);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

"Habib": قم بإجراء بحث عام غير حساس لحالة الأحرف عن

```
let text = "Abu Habib Al Husini *_ * haBib *_ * HABIB *_*";  
text.match(/hAbIB/gi);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ملحوظة

. فسيُرجع فقط المطابقة الأولى في النص (`match()`), (البحث العام) و إذا كان التعبير العادي لا يتضمن المُعدّل

## الدالة ( ) matchAll

. بإرجاع مكرر يحتوي على نتائج مطابقة نص مع نص (أو تعبير عادي) (`matchAll()`) تقوم الطريقة

مثال ↓↓ ✓

```
const iterator = text.matchAll("Habib");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

• وإلا فسيتم طرح خطأ، (g) إذا كانت المعلمة تعبيرًا عاديًا، فيجب تعيين العلامة العامة

✓ مثال ↓↓

```
const iterator = text.matchAll(/Habib/g);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

(i): إذا كنت تريد البحث غير حساس لحالة الأحرف، فيجب تعيين العلامة غير الحساسة

✓ مثال ↓↓

```
const iterator = text.matchAll(/Habib/gi);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظات

لا يعمل في إنترنت إكسبلورر (`matchAll()`)

## كيف معرفة نص البحث موجود أم لا

ترجع صحيحًا إذا كان النص موجود على قيم المحددة (`includes()`) ترجع الطريقة

`false` وإلا فإنه يعود

أمثلة

:"تحقق مما إذا كانت النص تتضمن "العالم

```
let text = "Abu Habib Al Husini *_*, welcome to the Arabic.";  
text.includes("Arabic");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

تحقق مما إذا كانت النص يتضمن "الكلمة المراد البحث عنها و . ابدأ من الموضع 12

```
let text = "Abu Habib Al Husini *_*, welcome to the Arabic.";
text.includes("Arabic", 12);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظات

حاله الاحرف حساسه `includes()`

Internet Explorer غير مدعوم في `includes()`

## طرق اخرى للتشيك على النصوص

كانت موجود ترجع `true` الطريقة إذا `startsWith()`

أمثلة

يعود صحيحا:

```
let text = "Abu Habib Al Husini *_*, welcome to the universe.";
text.startsWith("Husini");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

إرجاع كاذبة:

```
let text = "Abu Habib Al Husini *_*, welcome to the Arabic.";
text.startsWith("Arabic")
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يمكن تحديد موضع البدء للبحث

إرجاع كاذبة:

```
let text = "Abu Habib Al Husini *_*, welcome to the universe.";  
text.startsWith("Arabic", 5)
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

يعود صحيحا

```
let text = "Abu Habib Al Husini *_*, welcome to the universe.";  
text.startsWith("Arabic", 6)
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

## ملحوظات

مثل الدالة السابقة حاله الاحرف حساسه `startsWith()`

Internet Explorer غير مدعوم في `startsWith()`

## كيف البحث في نهاية النصوص

انتهت النصوص بقيمة محددة `true` تُرجع الطريقة إذا `endsWith()`

`false` وإلا فإنه يعود

أمثلة

"Al Husini ﷺ": تحقق مما إذا كانت النصوص تنتهي بـ

```
let text = "Abu Bakr Al-Siddiq";  
text.endsWith("Al Husini ﷺ");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

"Arabic" تحقق مما إذا كانت الأحرف الـ 11 الأولى من النص تنتهي بـ

```
let text = "Abu Habib Al Husini *_*, welcome to the Arabic.";  
text.endsWith("Arabic", 11);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## ملحوظات

مثل الدالة السابقة حاله حساسه `endsWith()`

## الاقباس الحر

تستخدم القيم الحرفية للنموذج علامات التجزئة الخلفية (``) بدلاً من علامات الاقتباس ("" ) لتحديد نص حر غير مقيد بقوانين جافاسكربت :

مثال ↓↓ ✓

```
let text = `Abu Habib Al Husini *_*!`;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

## اقتباسات داخل الاقتباس الحر

باستخدام الاقتباس الحر ، يمكنك استخدام علامات الاقتباس المفردة والمزدوجة داخل نص ونقل النصوص والرموز وكل شى بحرية بدون تطبيق قوانين جافاسكربت على النصوص :

✓ مثال ↓↓

```
let text = `He's often called "habib";
```

كتاب الكافي فى جافاسكربت الجزء الاول ابو حبيب الحسينى

## تابع الاقتباس الحر

:تسمح القيم الحرفية للاقتباس الحر بنصوص متعددة الأسطر

✓ مثال ↓↓

```
let text =  
`The quick  
brown fox  
jumps over  
the lazy dog`;
```

كتاب الكافي فى جافاسكربت الجزء الاول ابو حبيب الحسينى

## ملحوظة

تستطيع دمج التغيرات داخل الاقتباس الحر و توفر القيم الحرفية للاقتباس الحر طريقة سهلة لاستكمال المتغيرات والتعبيرات في نصوص

. تسمى هذه الطريقة استيفاء النص

:بناء الجملة هو

```
`${...}`
```

## مثال اخر لاستخدام الإقتباس الحر

كما ذكرنا تسمح القيم الحرفية للاقتباس الحر بالمتغيرات في النصوص

✓ مثال ↓↓

```
let firstName = "Habib";  
let lastName = "Al Husini 🍌🍌🍌";  
let text = `Welcome ${firstName}, ${lastName}!`;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

. يسمى الاستبدال التلقائي للمتغيرات بالقيم الحقيقية باستيفاء النص

## طريقة دمج اكواد داخل النصوص

تسمح القيم الحرفية للاقتباس الحر بالتعبيرات والاكواد في النصوص

✓ مثال ↓↓

```
let hAbiB3= 10;  
let VAT = 0.25;  
let total = `Total: ${((hAbiB3* (1 + VAT)).toFixed(2))}`;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

. يسمى الاستبدال التلقائي للتعبيرات بالقيم الحقيقية استيفاء النص



## قوالب HTML

✓ مثال ↓↓

```
let header = "Templates Literals";
let tags = ["template literals", "habib"];

let html = `<h2>${header}</h2><ul>`;
for (const x of tags) {
  html += `<li>${x}</li>`;
}

html += `</ul>`;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## دعم المتصفحات

تُهو مدعوم في جميع المتصفحات الحديثة

Internet Explorer غير مدعوم في **Template Literals**.

## كتاب المرجع الكامل في جاسكربت

:للحصول على مرجع الدوال كامل، انتقل إلى

. يحتوي المرجع على أوصاف وأمثلة لجميع خصائص النص وأساليبها

## طرق التعامل مع الأرقام

نعيد للتذكير تحتوي جافاسكربت على نوع واحد فقط من الأرقام. يمكن كتابة الأرقام مع أو بدون الكسور العشرية الانواع الاخرى لن تحتاج اليها فهي لمحترفي الجبر والرياضيات .

✓ مثال ↓↓

```
let x = 3.14; // A number with decimals
let y = 3; // A number without decimals
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

يمكن كتابة الأعداد الكبيرة جدًا أو الصغيرة جدًا باستخدام التدوين العلمي (الأس)

✓ مثال ↓↓

```
let x = 123e5; // 12300000
let y = 123e-5; // 0.00123
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تكون أرقام جافاسكريبت دائمًا عبارة عن نقطة عائمة او عشرية بحجم 64 بت

على عكس العديد من لغات البرمجة الأخرى، لا تحدد جافاسكريبت أنواعًا مختلفة من الأرقام، مثل الأعداد الصحيحة والقصيرة والطويلة والفاصلة العشرية وما إلى ذلك

الدولي IEEE 754 يتم دائمًا تخزين أرقام جافاسكريبت كرقم فاصلة عائمة مزدوجة المكون، وفقًا لمعيار

يقوم هذا التنسيق بتخزين الأرقام في 64 بت، حيث يتم تخزين الرقم (الكسر) في البتات من 0 إلى 51، والأس في البتات من 52 إلى 62، وتسجيل الدخول بت 63

## الأعداد الصحيحة

الأعداد الصحيحة (الأرقام التي لا تحتوي على نقطة أو تدوين أسّي) تصل دقتها إلى 15 رقمًا

✓ مثال ↓↓

```
let x = 999999999999999; // x will be 999999999999999
let y = 999999999999999; // y will be 10000000000000000
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ملحوظة

الحد الأقصى لعدد الكسور العشرية هو 17.

## الأعداد العشرية

حساب النقطة العشرية ليس دائمًا دقيقًا بنسبة 100

$$\text{let } x = 0.2 + 0.1;$$

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

لحل المشكلة أعلاه يساعد على الضرب والقسمة

$$\text{let } x = (0.2 * 10 + 0.1 * 10) / 10;$$

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## إضافة الأرقام والنصوص معا

!! تحذير

تستخدم جافاسكريبت عامل التشغيل + لكل من الإضافة والتسلسل.

تتم إضافة الأرقام، النصوص متنص

إذا قمت بإضافة رقمين، ستكون النتيجة رقما

✓ مثال ↓↓

```
let x = 10;  
let y = 20;  
let z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

: إذا قمت بإضافة سلسلتين، ستكون النتيجة نص

✓ مثال ↓↓

```
let x = "10";  
let y = "20";  
let z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

: إذا قمت بإضافة رقم ونص ، ستكون النتيجة نص

✓ مثال ↓↓

```
let x = 10;  
let y = "20";  
let z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

: إذا قمت بإضافة نص ورقم، ستكون النتيجة نص

✓ مثال ↓↓

```
let x = "10";  
let y = 20;  
let z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

: من الأخطاء الشائعة توقع أن تكون هذه النتيجة 30

✓ مثال ↓↓

```
let x = 10;  
let y = 20;  
let z = "The result is: " + x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

من الأخطاء الشائعة توقع أن تكون هذه النتيجة 102030

✓ مثال ↓↓

```
let x = 10;  
let y = 20;  
let z = "30";  
let result = x + y + z;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسينى

يعمل مترجم جافاسكربت من اليسار إلى اليمين

كلاهما رقمان  $x$  و  $y$  تمت إضافة أول  $20 + 10$  لأن

عبارة عن نص  $z$  ثم يتم ربط  $30 + "30"$  لأن

## النصوص الرقمية

يمكن أن تحتوي نصوص جافاسكربت على محتوى رقمي

```
let x = 100; // x is a number
```

```
let y = "100"; // y is a string
```

ستحاول جافاسكربت تحويل النصوص إلى أرقام في جميع العمليات الرقمية

هذا سيفي بالغرض

```
let x = "100";  
let y = "10";  
let z = x / y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

سيعمل هذا أيضًا

```
let x = "100";  
let y = "10";  
let z = x * y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

وهذا سوف يعمل

```
let x = "100";  
let y = "10";  
let z = x - y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

لكن هذا لن ينجح

```
let x = "100";  
let y = "10";  
let z = x + y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

في ال ✓ مثال ↓↓ الأخير، يستخدم جافاسكريبت عامل التشغيل + لتسلسل النصوص

## الكلمة NaN

هي كلمة محجوزة في جافاسكريبت تشير إلى أن الرقم ليس رقمًا قانونيًا NaN

(ليس رقمًا) NaN ستؤدي محاولة إجراء العمليات الحسابية باستخدام نص غير رقمية إلى

✓ مثال ↓↓

```
let x = 100 / "Abu Habib";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

نومع ذلك، إذا كانت النص تحتوي على قيمة رقمية، فستكون النتيجة رقمًا

✓ مثال ↓↓

```
let x = 100 / "10";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

لمعرفة ما إذا كانت القيمة رقمًا أم لا (**isNaN()**) يمكنك استخدام وظيفة جافاسكربت العامة

✓ مثال ↓↓

```
let x = 100 / "Abu Habib";  
isNaN(x);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

**NaN:** في عملية رياضية، ستكون النتيجة أيضًا NaN إذا استخدمت NaN. احترس من

✓ مثال ↓↓

```
let x = NaN;  
let y = 5;  
let z = x + y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

**NaN5:** أو قد تكون النتيجة نص مثل

✓ مثال ↓↓

```
let x = NaN;  
let y = "5";  
let z = x + y;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

**number** يعود **typeof NaN** : هو رقم **NaN**

✓ مثال ↓↓

```
typeof NaN;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## القيمة الـ نهاية

هي القيمة التي سيرجعها جافاسكريبت إذا قمت بحساب رقم خارج أكبر رقم ممكن (**-Infinity** أو **Infinity**)

✓ مثال ↓↓

```
let myNumber = 2;  
// Execute until Infinity  
while (myNumber != Infinity) {  
  myNumber = myNumber * myNumber;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

**Infinity** القسمة على 0 (صفر) تولد أيضاً

✓ مثال ↓↓

```
let x = 2 / 0;  
let y = -2 / 0;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

**number** يعود **typeof Infinity** : هو رقم **Infinity**.



✓ مثال ↓↓

**typeof Infinity;**

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## السداسي عشري

X. تفسر جافاسكريبت الثوابت الرقمية على أنها أرقام ست عشرية إذا كانت مسبوقه بـ 0

✓ مثال ↓↓

**let x = 0xFF;**

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

لا تكتب أبدأ رقمًا بصفر بادئ (مثل 07)

تفسر بعض إصدارات جافاسكريبت الأرقام على أنها ثماني إذا كانت مكتوبة بصفر بادئ

افترضيًا، تعرض جافاسكريبت الأرقام على هيئة أرقام عشرية ذات أساس 10

الطريقة لإخراج الأرقام من الأساس 2 إلى الأساس 36 **toString()** ولكن يمكنك استخدام

النظام الست عشري هو الأساس 16 . العدد العشري هو الأساس 10 . الثماني هو قاعدة 8 . ثنائي هو قاعدة 2

✓ مثال ↓↓

```
let myNumber = 32;  
myNumber.toString(32);  
myNumber.toString(16);  
myNumber.toString(12);  
myNumber.toString(10);  
myNumber.toString(8);  
myNumber.toString(2);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## أرقام كائنات

عادةً ما تكون أرقام جافاسكربت عبارة عن قيم بدائية تم إنشاؤها من القيم الحرفية

```
let x = 123;
```

**new**: ولكن يمكن أيضًا تعريف الأرقام ككائنات باستخدام الكلمة المحجوزة

```
let y = new Number(123);
```

مثال ↓↓

```
let x = 123;
```

```
let y = new Number(123);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## لا ينصح باستخدامها

لا تقم بإنشاء كائنات رقمية

المحجوزة على تعقيد التعليمات البرمجية وإبطاء سرعة التنفيذ **new** تعمل الكلمة

:يمكن أن تؤدي الكائنات الرقمية إلى نتائج غير متوقعة

:متساويان لا و X، عند استخدام == المعامل

```
let x = 500;
```

```
let y = new Number(500);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

:ليسا متساويين لا و X، عند استخدام === المعامل

```
let x = 500;  
let y = new Number(500);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

صحيحة أو خاطئة؟  $(x == y)$  و  $(x === y)$ . لاحظ الفرق بين

صحيحة أو خاطئة؟  $(x == y)$

```
let x = new Number(500);  
let y = new Number(500);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

صحيحة أو خاطئة؟  $(x === y)$

```
let x = new Number(500);  
let y = new Number(500);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

**false** دائماً ما تؤدي المقارنة بين كائنين في جافاسكربت إلى إرجاع

## امثلة على BigInt

تُستخدم متغيرات جافاسكربت لتخزين قيم أعداد صحيحة كبيرة جداً بحيث لا يمكن تمثيلها بواسطة جافاسكربت **BigInt** عادي **Number**.

## مكون الأعداد الصحيحة

تعد أعداد جافاسكربت الصحيحة دقيقة حتى 15 رقمًا فقط

## مكون عدد صحيح

```
let x = 9999999999999999;  
let y = 9999999999999999;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

(IEEE 754 معيار) في جافاسكربت، يتم تخزين جميع الأرقام بتنسيق الفاصلة العشرية 64 بت

باستخدام هذا المعيار، لا يمكن تمثيل الأعداد الصحيحة الكبيرة بشكل دقيق وسيتم تقريبها

:ولهذا السبب، يمكن لـ جافاسكربت تمثيل الأعداد الصحيحة بشكل آمن فقط

حتى 9007199254740991 + (2 53) - 1

و

9007199254740991 - (2 53) - 1

القيم الصحيحة خارج هذا النطاق تفقد المكون

## كيفية إنشاء نوع البيانات BigInt

BigInt(): بنهاية عدد صحيح أو قم باستدعاء n قم بإلحاق، لإنشاء ملف

أمثلة

```
let x = 9999999999999999;  
let y = 9999999999999999n;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

```
let x = 1234567890123456789012345n;  
let y = BigInt(1234567890123456789012345)
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## تابع: BigInt

": عدد كبير "BigInt" typeofa جافاسكربت

✓ مثال ↓↓

```
let x = BigInt(9999999999999999);  
let type = typeof x;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

**BigInt** (بعد **Number**) هو نوع البيانات الرقمية الثاني في جافاسكربت.

إجمالي عدد أنواع البيانات المدعومة في جافاسكربت هو **8** مع **BigInt**:

1. نص
2. رقم
3. كبير
4. منطقي
5. غير محدد
6. فارغ
7. رمز
8. كائن

## مشغل BigInt

**BigInt** يمكن أيضاً استخدام العوامل التي يمكن استخدامها في جافاسكربت على **Number**.

✓ مثال ↓↓ الضرب

```
let x = 9007199254740995n;  
let y = 9007199254740995n;  
let z = x * y;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظات

غير مسموح به (تحويل النوع يفقد المعلومات في الارقام الكبيرة) **Number** **a** و **BigInt** **a** الحساب بين

(ليس له عرض ثابت) **BigInt** لا يمكن إجراء التحويل الأيمن غير الموقع بهذه العلامة (<<<) على



MAX\_SAFE\_INTEGER ✓ مثال ↓↓

```
9007199254740992 === 9007199254740993; // is true !!!
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## دعم المتصفحات

مدعوم في جميع المتصفحات منذ سبتمبر 2020 BigInt

## الحد الأدنى والحد الأقصى للأعداد الصحيحة الآمنة

خصائص الحد الأقصى والحد الأدنى إلى كائن الرقم في ES6 أضاف

- MAX\_SAFE\_INTEGER
- MIN\_SAFE\_INTEGER

MAX\_SAFE\_INTEGER ✓ مثال ↓↓

```
let x = Number.MAX_SAFE_INTEGER;
```

```
let x = Number.MIN_SAFE_INTEGER;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## طرق او دوال الأرقام

أيضاً طريقتين جديدتين إلى كائن الرقم ES6 أضاف

- Number.isInteger()
- Number.isSafeInteger()

## طريقة Number.isInteger()

كانت الوسيلة عددًا صحيحًا true تُرجع الطريقة إذا Number.isInteger()

## ✓ مثال `isInteger()`: ↓↓

```
Number.isInteger(10);  
Number.isInteger(10.5);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## طريقة `Number.isSafeInteger()`

العدد الصحيح الآمن هو عدد صحيح يمكن تمثيله بمكون كرقم مزدوج المكون

كانت الوسيطة عددًا صحيحًا آمنًا `true` تُرجع الطريقة إذا `Number.isSafeInteger()`

## ✓ هو `SafeInteger()` مثال ↓↓

```
Number.isSafeInteger(10);  
Number.isSafeInteger(12345678901234567890);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

الأعداد الصحيحة الآمنة هي جميع الأعداد الصحيحة من  $(-2^{53} - 1)$  إلى  $(2^{53} - 1)$ .  
هذا آمن: 9007199254740991. هذا غير آمن: 9007199254740992

## طرق رقم جافاسكربت

## طرق ارقام جافاسكربت

يمكن استخدام طرق الأرقام هذه على جميع أرقام جافاسكربت

## `() toString` طريقة.

• بإرجاع رقم كنص `toString()` تقوم الطريقة



يمكن استخدام جميع طرق الأرقام على أي نوع من الأرقام (الأحرف الحرفية أو المتغيرات أو التعبيرات)

✓ مثال ↓↓

```
let x = 123;  
x.toString();  
(123).toString();  
(100 + 23).toString();
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## toExponential() طريقة

تقوم بإرجاع نص تحتوي على رقم مقرب ومكتوب باستخدام التدوين الأسّي (`toExponential()`)

تحدد المعلمة عدد الأحرف خلف العلامة العشرية

✓ مثال ↓↓

```
let x = 9.656;  
x.toExponential(2);  
x.toExponential(4);  
x.toExponential(6);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

المعلمة اختيارية. إذا لم تحده، فلن تقوم جافاسكريبت بتقريب الرقم

## toFixed() طريقة

تُرجع نصوصة تحتوي على الرقم المكتوب بعدد محدد من الكسور العشرية (`toFixed()`)

✓ مثال ↓↓

```
let x = 9.656;  
x.toFixed(0);  
x.toFixed(2);  
x.toFixed(4);  
x.toFixed(6);
```

مثال  $\Downarrow\Downarrow$  ي للعمل بالمال  $\checkmark$  `toFixed(2)`

## طريقة `toPrecision()`

إرجاع نص برقم مكتوب بطول محدد `toPrecision()`

مثال  $\Downarrow\Downarrow$   $\checkmark$

```
let x = 9.656;  
x.toPrecision();  
x.toPrecision(2);  
x.toPrecision(4);  
x.toPrecision(6);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## طرق اخرى لجلب القيمة

إرجاع القيمة كرقم `valueOf()`

مثال  $\Downarrow\Downarrow$   $\checkmark$

```
let x = 123;  
x.valueOf();  
(123).valueOf();  
(100 + 23).valueOf();
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

`(typeof = object)` أو كائنًا `(typeof = number)` في جافاسكريبت، يمكن أن يكون الرقم قيمة أولية

الطريقة داخليًا في جافاسكريبت لتحويل كائنات الأرقام إلى قيم أولية `valueOf()` يتم استخدام هذه

لا يوجد سبب لاستخدامه في التعليمات البرمجية

`valueOf()` و `toString()` تحتوي جميع أنواع بيانات جافاسكريبت على طريقة

## لتحويل المتغيرات إلى أرقام

هناك ثلاث طرق جافاسكربت يمكن استخدامها لتحويل متغير إلى رقم

## الطريقة الأولى Number()

لتحويل متغيرات جافاسكربت إلى أرقام **Number()** يمكن استخدام الطريقة

✓ مثال ↓↓

```
Number(true);  
Number(false);  
Number("10");  
Number(" 10");  
Number("10 ");  
Number(" 10 ");  
Number("10.33");  
Number("10,33");  
Number("10 33");  
Number("Habib");
```

كتاب الكافي في جافاسكربت الجزء الأول أبو حبيب الحسيني

فسيتم إرجاع (ليس رقمًا) **NaN**، إذا تعذر تحويل الرقم

## كيف تحويل التاريخ Number()

. يمكن أيضًا تحويل التاريخ إلى رقم باستخدام **Number()**

✓ مثال ↓↓

```
Number(new Date("1970-01-01"))
```

كتاب الكافي في جافاسكربت الجزء الأول أبو حبيب الحسيني

•إرجاع عدد المللي ثانية منذ **Number()**1.1.1970 تقوم الطريقة

:عدد المللي ثانية بين 1970-01-02 و1970-01-01 هو 86400000

✓ مثال ↓↓

```
Number(new Date("1970-01-02"))
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

✓ مثال ↓↓

```
Number(new Date("2017-09-30"))
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## الطريقة الثانية () parseInt

:ياخذ نص ويعيد عددا صحيحا. المساحات مسموحة. يتم إرجاع الرقم الأول فقط **parseInt()**

✓ مثال ↓↓

```
parseInt("-10");  
parseInt("-10.33");  
parseInt("10");  
parseInt("10.33");  
parseInt("10 20 30");  
parseInt("10 years");  
parseInt("years 10");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

.فسيتم إرجاع (ليس رقماً) **NaN**، إذا تعذر تحويل الرقم

## الطريقة الثالثة () parseFloat

:ياخذ نص ويعيد رقماً. المساحات مسموحة. يتم إرجاع الرقم الأول فقط **parseFloat()**

✓ مثال ↓↓

```
parseFloat("10");  
parseFloat("10.33");  
parseFloat("10 20 30");  
parseFloat("10 years");  
parseFloat("years 10");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

فسيتم إرجاع (ليس رقمًا) NaN، إذا تعذر تحويل الرقم

## شرح اهم دوال الأرقام

: تنتمي أساليب الكائن هذه إلى كائن أو كلاس الرقم

طريقة	وصف
Number.isInteger()	يُرجع صحيحًا إذا كانت الوسيطة عددًا صحيحًا
Number.isSafeInteger()	يُرجع صحيحًا إذا كانت الوسيطة عددًا صحيحًا آمنًا
Number.parseFloat()	تحويل نص إلى رقم
Number.parseInt()	تحويل نص إلى عدد صحيح

## لا يمكن استخدام الأساليب الرقمية في المتغيرات

. تنتمي طرق الأرقام المذكورة أعلاه إلى كائنا رقم جافاسكريبت

. `Number.isInteger()` لا يمكن الوصول إلى هذه الأساليب في حالة المتغير وإلا ستؤدي إلى

: إلى حدوث خطأ X حيث `X.isInteger()` سيؤدي استخدام

**TypeError X.isInteger is not a function.**

## دالة (`Number.isInteger()`)

كانت الوسيطة عددًا صحيحًا **true** تُرجع الطريقة إذا (`Number.isInteger()`)

✓ مثال ↓↓

```
Number.isInteger(10);
```

```
Number.isInteger(10.5);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## دالة (`Number.isSafeInteger()`)

العدد الصحيح الآمن هو عدد صحيح يمكن تمثيله بمكون كرقم مزدوج المكون

كانت الوسيطة عددًا صحيحًا آمنًا **true** تُرجع الطريقة إذا (`Number.isSafeInteger()`)

✓ مثال ↓↓

```
Number.isSafeInteger(10);
```

```
Number.isSafeInteger(12345678901234567890);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

الأعداد الصحيحة الآمنة هي جميع الأعداد الصحيحة من  $(-2^{53} - 1)$  إلى  $(2^{53} - 1)$

هذا آمن: 9007199254740991. هذا غير آمن: 9007199254740992

## دالة (`Number.parseFloat()`)

ياخذ نص ويعيد رقمًا (`Number.parseFloat()`)

:المساحات مسموحة. يتم إرجاع الرقم الأول فقط

✓ مثال ↓↓

```
Number.parseFloat("10");  
Number.parseFloat("10.33");  
Number.parseFloat("10 20 30");  
Number.parseFloat("10 years");  
Number.parseFloat("years 10");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

فسيتم إرجاع (ليس رقمًا) NaN، إذا تعذر تحويل الرقم

## ملحوظة

`Number.parseInt()` و `Number.parseFloat()`. الطريقتان متشابهتان في التنفيذ

`parseInt()` و `parseFloat()`

والغرض من ذلك هو تقسيم الوحدات العامة (لتسهيل استخدام نفس كود جافاسكريبت خارج المتصفح)

## الدالة `Number.parseInt()`

يأخذ نص ويعيد عددا صحيحا (`Number.parseInt()`)

:المساحات مسموحة. يتم إرجاع الرقم الأول فقط

✓ مثال ↓↓

```
Number.parseInt("-10");  
Number.parseInt("-10.33");  
Number.parseInt("10");  
Number.parseInt("10.33");  
Number.parseInt("10 20 30");  
Number.parseInt("10 years");  
Number.parseInt("years 10");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

فسيتم إرجاع (ليس رقمًا) NaN، إذا تعذر تحويل الرقم

## كتاب المرجع الكامل لجافاسكربت

للحصول على مرجع لحوال الأرقام في جافا سكربت كاملة، قم بزيارة موقع القافلة المهنية بمجرد التسجيل ستجد جميع كتب ابو حبيب الحسيني كاملة والغير موجودة في المكتبات الأخرى: تحميل مجاني

يحتوي المرجع على أوصاف وأمثلة لجميع خصائص وأساليب و دوال جافاسكربت بلا استثناء

### خصائص الأرقام

Property	وصف
EPSILON	JS الفرق بين 1 وأصغر رقم
MAX_VALUE	أكبر عدد ممكن في جافاسكربت
MIN_VALUE	أصغر رقم ممكن في جافاسكربت
MAX_SAFE_INTEGER	الحد الأقصى لعدد صحيح آمن (1 - 253)
MIN_SAFE_INTEGER	الحد الأدنى لعدد صحيح آمن - (1 - 253)
POSITIVE_INFINITY	إنفينيتي (يرجع الفأض)
NEGATIVE_INFINITY	اللانهاية السالبة (التي تم إرجاعها عند الفأض)
NaN	"قيمة" ليست رقمًا

### خاصية إبسيلون

يمثل الفرق بين 1 وأصغر رقم الفاصلة العشرية `Number.EPSILON`

مثال ↓↓ ✓

```
let x = Number.EPSILON;
```



## ملحوظة

**ES6** هي ميزة **Number.EPSILON**.

أنها لا تعمل في إنترنت إكسبلورر.

## خاصية MAX\_VALUE

هو ثابت يمثل أكبر عدد ممكن في جافاسكربت **Number.MAX\_VALUE**.

✓ مثال ↓↓

```
let x = Number.MAX_VALUE;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## لا يمكن استخدام خصائص الأرقام في المتغيرات

. تنتمي خصائص الأرقام إلى كائن الأرقام

**Number.MAX\_VALUE** لا يمكن استخدام هذه الخاصية إلا كنوع البيانات الأرقام الكبيرة بيج انت

**undefined**: متغير أو قيمة، سيتم إرجاع **x** حيث **x.MAX\_VALUE** باستخدام

✓ مثال ↓↓

```
let x = 6;  
x.MAX_VALUE
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## خاصية MIN\_VALUE

هو ثابت يمثل أقل رقم ممكن في جافاسكربت `Number.MIN_VALUE`.

✓ مثال ↓↓

```
let x = Number.MIN_VALUE;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## خاصية MAX\_SAFE\_INTEGER

يمثل الحد الأقصى لعدد صحيح آمن في جافاسكربت `Number.MAX_SAFE_INTEGER`.

هو `Number.MAX_SAFE_INTEGER(1 - 2 53)`.

✓ مثال ↓↓

```
let x = Number.MAX_SAFE_INTEGER;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## خاصية MIN\_SAFE\_INTEGER

يمثل الحد الأدنى لعدد صحيح آمن في جافاسكربت `Number.MIN_SAFE_INTEGER`.

هو `Number.MIN_SAFE_INTEGER(1 - 2 53)`.

✓ مثال ↓↓

```
let x = Number.MIN_SAFE_INTEGER;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## خاصية POSITIVE\_INFINITY

✓ مثال ↓↓

```
let x = Number.POSITIVE_INFINITY;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

POSITITIVE\_INFINITY يتم إرجاعها عند الفائص

```
let x = 1 / 0;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## خاصية NEGATIVE\_INFINITY

✓ مثال ↓↓

```
let x = Number.NEGATIVE_INFINITY;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

NEGATIVE\_INFINITY يتم إرجاعها عند الفائص

```
let x = -1 / 0;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## شرح NaN

هي كلمة جافاسكربت محجوزة تعنى انه ليس رقمًا قانونيًا NaN

### أمثلة

```
let x = Number.NaN;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

: (ليس رقمًا) NaN ستؤدي محاولة إجراء العمليات الحسابية باستخدام نص غير رقمية إلى

```
let x = 100 / "Abu Habib";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## المصفوفات

:المصفوفة عبارة عن متغير خاص يمكنه الاحتفاظ بأكثر من قيمة واحدة

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## لماذا استخدام المصفوفات؟

إذا كانت لديك قائمة بالعناصر (قائمة بأسماء ، على سبيل ال ✓ مثال ↓↓)، فقد يبدو تخزين في متغيرات فردية كما يلي

```
let $_Habib1 = "Habib";
```

```
let $_Habib2 = "Hamza";
```

```
let $_Habib3 = "Abu Habib Al-Husini";
```

كما ترى في الأعلى هذه ثلاث متغيرات لها قيم ماذا لو كنت تريد التنقل والعثور على قيمة معينة بين هذه ، المتغيرات؟ وماذا لو لم يكن لديك 3 متغيرات فقط بل مليون ماذا ستفعل؟

الحل هو المصفوفة

يمكن للمصفوفة أن تحتوي على العديد من القيم تحت اسم واحد، ويمكنك الوصول إلى تخزين كبير جدا لتصبح قاعدة بيانات كبيرة من القيم ليسهل الوصول إليها عن طريق الإشارة إلى رقم الفهرس او بطرق اخرى كثيرة جدا فقد اعدت لذلك

## كيف إنشاء مصفوفة

يعد استخدام المصفوفة الحرفية أسهل طريقة لإنشاء مصفوفة جافاسكربت

بناء الجملة:

```
const array_name = [item1, item2, ...];
```

من الممارسات الشائعة الإعلان عن المصفوفات باستخدام الكلمة المحجوزة كونسنت و ليست اجباره

✓ مثال ↓↓

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

المسافات وفواصل الأسطر ليست مهمة. يمكن أن يمتد الإعلان على عدة أسطر

✓ مثال ↓↓

```
const HaBiB= [  
  "Habib",  
  "Hamza",  
  "Abu Habib Al-Husini"  
];
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

يمكنك أيضًا إنشاء مصفوفة، ثم توفير العناصر

✓ مثال ↓↓

```
const HaBiB= [];  
HaBiB[0]= "Habib";  
HaBiB[1]= "Hamza";  
HaBiB[2]= "Abu Habib Al-Husini";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## استخدام الكلمة كونست

يقوم ال ✓ مثال ↓↓ التالي أيضًا بإنشاء مصفوفة، وتعيين قيم لها شاهد

✓ مثال ↓↓

```
const HaBiB= new Array("Habib", "Hamza", "Abu Habib Al-Husini");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ال ✓ مثال ↓↓ ان أعلاه يفعلان نفس الشيء تمامًا

ليست هناك حاجة للاستخدام **new Array()**.

من أجل البساطة وسهولة القراءة وسرعة التنفيذ، استخدم الطريقة الحرفية للمصفوفة

## الوصول إلى عناصر المصفوفة

: يمكنك الوصول إلى عنصر المصفوفة بالرجوع إلى رقم الفهرس

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];  
let $_Habib= HaBiB[0];
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

ملحوظة: فهرس المصفوفات تبدأ بالرقم 0

هو العنصر الأول. [1] هو العنصر الثاني [0]

## كيف تغيير عناصر المصفوفة

**HaBiB**: يغير هذا قيمة العنصر الأول في

```
HaBiB[0] = "Omar *";
```

مثال ↓↓ ✓

```
const HaBiB= ["Habib", "Samir", "Abu Habib Al-Husini"];  
HaBiB[0] = "Hamza *";
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## الوصول إلى المصفوفة بكاملة

: باستخدام جافاسكربت، يمكن الوصول إلى المصفوفة بكامل من خلال الإشارة إلى اسم المصفوفة

✓ مثال ↓↓

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];  
document.getElementById("Habib").innerHTML = HaBiB;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## المصفوفات هي كائنات

المصفوفات هي نوع خاص من الكائنات. او البيانات يقوم عامل التشغيل في جافاسكربت بإرجاع "كائن" للمصفوفات او العنصر .

: يُرجع `hAbiB[0]`، ↓↓، تستخدم المصفوفات الأرقام للوصول إلى "عناصرها". في هذا ال ✓ مثال

مثال

```
const hAbiB = ["Habib", "Al Husini 😊😊😊", 406, 4006, 4566];
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

: يُرجع الناتج ب حبيب `hAbiB.firstName`، ↓↓، تستخدم الكائنات الأسماء للوصول إلى "أعضائها". في هذا ال ✓ مثال

هدف:

```
const hAbiB = {firstName:"Habib", lastName:"Al Husini 😊😊😊", age:40};
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## يمكن أن تكون عناصر المصفوفة كائنات

يمكن أن تكون متغيرات جافاسكربت كائنات. المصفوفات هي أنواع خاصة من الكائنات

وبسبب هذا، يمكن أن يكون لديك متغيرات من أنواع مختلفة في نفس المصفوفة

يمكن أن يكون لديك كائنات في مصفوفة . يمكن أن يكون لديك وظائف في مصفوفة . يمكن أن يكون لديك مصفوفات في مصفوفة :



```
Habib_Array[0] = Date.now;  
Habib_Array[1] = Husini;  
Habib_Array[2] = myHaBiB;
```

## خصائص المصفوفة وطرقها

تتضمن القوة الحقيقية لمصفوفات جافاسكريبت في خصائص وأساليب المصفوفة المضمنة

```
HaBiB.length // Returns the number of elements  
HaBiB.sort() // Sorts the array
```

سيتم تناول طرق المصفوفة في الفصول التالية.

## كيف معرفة طول المصفوفة

المصفوفة بإرجاع طول المصفوفة (عدد عناصر المصفوفة) **length** تقوم خاصية

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
let length = HaBiB.length;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

سيرجع هذا المثال برقم يزيد بواحد عن رقم المصفوفة لان المصفوفة تبدأ من الصفر

## كيف الوصول إلى العنصر الأول

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
let fruit = HaBiB[0];
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف الوصول إلى العنصر الأخير

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
let fruit = HaBiB[HaBiB.length - 1];
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف عرض جميع العناصر

حلقة **for** إحدى الطرق للتكرار عبر المصفوفة هي استخدام

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
let fLen = HaBiB.length;  
  
let text = "<ul>";  
for (let i = 0; i < fLen; i++) {  
  text += "<li>" + HaBiB[i] + "</li>";  
}  
text += "</ul>";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

: هذه الوظيفة **Array.forEach()** يمكنك أيضاً استخدامها وستحصل على نفس الناتج

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
  
let text = "<ul>";  
HaBiB.forEach(Husini);  
text += "</ul>";  
  
function Husini(value) {  
  text += "<li>" + value + "</li>";  
}
```

## كيف إضافة عناصر في نهاية المصفوفة

الطريقة (**push()**) أسهل طريقة لإضافة عنصر جديد إلى مصفوفة هي استخدام

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib"];  
HaBiB.push("Osama"); // Adds a new element (Osama) to HaBiB
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

هذه الخاصية **length** يمكن أيضًا إضافة عنصر جديد إلى نهاية المصفوفة باستخدامها شاهد

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "HamZA", "Abu Habib"];  
HaBiB[HaBiB.length] = "Osama"; // Adds "Osama" to HaBiB
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

! تحذير

يمكن أن تؤدي إضافة عناصر ذات فهارس عالية إلى إنشاء "فجوات" غير محددة في المصفوفة والافضل ان تترك الترتيب التلقائي يأخذ مجراه و اتباع الدوال التي سنشرحها للحفاظ على المفاتيح ويجب عمل ملف مستقل للمصفوفة اذا كانت كبيرة : الحجم لان الكمبيوتر سيعتبرها قاعدة بيانات

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", " * ", "Abu Habib"];  
HaBiB[6] = "Osama"; // Creates undefined "holes" in HaBiB
```

هنا اصبحت المصفوفة تحتوي على فجوات بين رقم خمسة ورقم ثلاثة والا استخدم الكائنات لتحكم اكثر في المفاتيح

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المصفوفات الترابطية

تدعم العديد من لغات البرمجة المصفوفات ذات الفهارس المسماة.  
تسمى المصفوفات ذات الفهارس المسماة المصفوفات الترابطية (أو التجزئة).  
لا تدعم جافاسكربت المصفوفات ذات الفهارس المسماة.  
في جافاسكربت، تستخدم المصفوفات دائماً فهرس مرقمة  
البديل لها هي الكائنات سنتناولها في دروس قادمة

✓ مثال ↓↓

```
const hAbiB = [];  
hAbiB[0] = "Habib";  
hAbiB[1] = "Al Husini ﷺ";  
hAbiB[2] = 46;  
hAbiB.length; // Will return 3  
hAbiB[0]; // Will return "Habib"
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

!! تحذير

إذا كنت تستخدم فهرسًا مسماة، فسوف تقوم جافاسكربت بإعادة تعريف المصفوفة لكائن ما

بعد ذلك، ستؤدي بعض أساليب وخصائص المصفوفة إلى نتائج غير صحيحة لأن المفاتيح أو الفهارس ستتغير

✓ مثال ↓↓↓:

```
const hAbiB = {};  
hAbiB["firstName"] = "Habib";  
hAbiB["lastName"] = "Al Husini ﷺ";  
hAbiB["age"] = 46;  
hAbiB.length; // Will return 0  
hAbiB[0]; // Will return undefined
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## الفرق بين المصفوفات والكائنات

. في جافاسكريبت، تستخدم المصفوفات فهرس مرقمة

. في جافاسكريبت، تستخدم الكائنات الفهارس المسماة وهي أكثر فاعلية

. المصفوفات هي نوع خاص من الكائنات، مع فهرس مرقمة لا تقبل نصوص في المفاتيح

## متى تستخدم المصفوفات و متى تستخدم الكائنات

- في الغالب ننصح باستخدام الكائنات عوضاً عن المصفوفات ولكن في بعض الأمور ستطر إلى استخدام المصفوفات
- يجب عليك استخدام الكائنات عندما تريد أن تكون أسماء العناصر عبارة عن نصوص (نص)
- يجب عليك استخدام المصفوفات عندما تريد أن تكون أسماء العناصر أرقاماً

## كيف استخدام new Array()

. `new Array()` تحتوي جافاسكريبت على مُنشئ مصفوفة مدمج

.ولكن يمكنك استخدامها بأمان `[]` بدلاً من ذلك

:تقوم هاتان العبارتان المختلفتان بإنشاء مصفوفة فارغة جديدة تسمى `array`

```
const hAbIB = new Array();  
const hAbIB = [];
```

:تقوم هاتان العبارتان المختلفتان بإنشاء مصفوفة جديدة تحتوي على 6 أرقام

```
const hAbIB = new Array(40, 100, 1, 5, 25, 10, 100, 1, 5, 25, 10);  
const hAbIB = [40, 100, 1, 5, 25, 10, 100, 1, 5, 25, 10, 0, 1, 5, 25, 10];
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

:المحجوزة بعض النتائج غير المتوقعة `new` يمكن أن تنتج الكلمة

```
// Create an array with three elements:  
const hAbIB = new Array(40, 100, 1);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

```
// Create an array with two elements:
```

```
const hAbIB = new Array(40, 100);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

```
// Create an array with one element ???
```

```
const hAbIB = new Array(40);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## خطأ شائع

```
const hAbIB = [40];
```

هذا وصول الى العنصر رقم اربعة واربعين

ليس هو نفسه كما يلي:

```
const hAbIB = new Array(40);
```

انما هذا انشاء مصفوفة من اربعة واربعين عنصر

```
// Create an array with one element:
```

```
const hAbIB = [40];
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

```
// Create an array with 40 undefined elements:
```

```
const hAbIB = new Array(40);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## كيفية التعرف على المصفوفة

السؤال الشائع هو: كيف أعرف ما إذا كان المتغير مصفوفة؟

:"object" يُرجع **typeof** تكمن المشكلة في أن عامل تشغيل جافاسكربت

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib"];  
let type = typeof HaBiB;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

بإرجاع كائن لأن مصفوفة جافاسكربت عبارة عن كائن `typeof` يقوم عامل التشغيل

## الحل: 1

`Array.isArray()` لحل هذه المشكلة، حدد (جافاسكربت 2009) طريقة جديدة

```
Array.isArray(HaBiB);
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## الحل: 2

إذا تم إنشاء كائن بواسطة مُنشئ معين `true` القيمة `instanceof` يُرجع العامل

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib"];
```

```
HaBiB instanceof Array;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

# طرق المصفوفات

## أولاً تحويل المصفوفات إلى نصوص

بتحويل مصفوفة إلى نص من قيم المصفوفة (مفصولة بفواصل) `toString()` تقوم طريقة جافاسكربت

مثال ↓↓↓ ✓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
document.getElementById("Habib").innerHTML = HaBiB.toString();
```

نتيجة:

## كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

• أيضًا بدمج كافة عناصر المصفوفة في نص `join()` تقوم الطريقة  
:ولكن بالإضافة إلى ذلك يمكنك تحديد الفاصل `toString()` إنه يتصرف تمامًا مثل

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
document.getElementById("Habib").innerHTML = HaBiB.join(" * ");
```

نتيجة:

Al-Husini \* Osman \* Abu Habib \* Abu Zedan

## كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

### تعديل المصفوفات

• عند العمل مع المصفوفات، يكون من السهل إزالة العناصر وإضافة عناصر جديدة.  
• لهذا ما هو الفرع والدفع  
• إخراج العناصر من المصفوفة، أو دفع العناصر إلى مصفوفة

### كيف ازالة العنصر الاخير من المصفوفة

• إزالة العنصر الأخير من المصفوفة `pop()` تقوم الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.pop();
```

## كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى



:"القيمة التي "برزت" **pop()** تُرجع الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
let fruit = HaBiB.pop();
```

سيتم إزالة ابو حمزا لانه العنصر الاخير

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف اضافة عنصر جديد في النهاية

:عنصرًا جديدًا إلى المصفوفة (في النهاية) **push()** تضيف الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.push("Hamza");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

:بإرجاع طول المصفوفة الجديد **push()** تقوم الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
let length = HaBiB.push("Hamza");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## اساليب الحذف

.بمماثلة تعديل، ولكنه يعمل على العنصر الأول بدلاً من العنصر الأخير **Shifting** يعتبر

## كيف حذف العنصر الأول من المصفوفة

بإزالة عنصر المصفوفة الأول وإزاحة مفاتيح جميع العناصر الأخرى إلى فهرس أقل (`shift()`) تقوم الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.shift();
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

"القيمة التي تم إزاحتها للخارج (`shift()`) تُرجع الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
let fruit = HaBiB.shift();
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

## كيف اضافة عنصر الى البداية والغاء الإزاحة للمفاتيح

عنصرًا جديدًا إلى المصفوفة (في البداية)، وتقوم بإلغاء إزاحة العناصر القديمة (`unshift()`) تضيف الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Abu Habib", "Abu Habib", "Abu Zedan"];  
HaBiB.unshift("Osama");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

بإرجاع طول المصفوفة الجديد (`unshift()`) تقوم الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.unshift("Osama");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

## كيف استبدال قيمة عنصر معين

: يتم الوصول إلى عناصر المصفوفة باستخدام رقم الفهرس الخاص بها

تبدأ فهرس المصفوفات بـ 0

هو عنصر المصفوفة الأول [0]

هو الثاني [1]

... هو الثالث [2]

مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB[0] = "Hamza";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف استخدام مجموع العناصر للإضافة

:طريقة سهلة لإلحاق عنصر جديد بالمصفوفة **length** توفر الخاصية

مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib"];  
HaBiB[HaBiB.length] = "Hamza";
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## حذف عناصر من المصفوفة

### ! تحذير

الكثير يستخدم هذا الاجراء ويظن ان العنصر تم حذفه بالكامل ولكن يمكن حذف عناصر المصفوفة باستخدام عامل تشغيل **delete** وستبقى مكان العنصر موجودة في المصفوفة فارغ لا يوجد بها قيمة تستطيع استدعئه من جديد والحل هنا

استخدم بدلاً من ذلك **Shift()** أو **pop()** للحذف التام.

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
delete HaBiB[0];
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف دمج المصفوفات

بإنشاء مصفوفة جديدة عن طريق دمج المصفوفات الموجودة (**concat()**) تقوم الطريقة

✓ مثال ↓↓ (دمج مصفوفتين)

```
const HABIB$ = ["zedan", "Braa"];  
const HuSin$ = ["Emil", "MoHamed", "Ahmid"];  
const Habib_myChildren = HABIB$.concat(HuSin$);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

لا تغير الطريقة المصفوفات الموجودة . يقوم دائماً بإرجاع مصفوفة جديدة (**concat()**)

أبي عدد من وسيطات المصفوفة (**concat()**) يمكن أن تأخذ الطريقة

مثال ↓↓ (دمج ثلاث مصفوقات) ✓

```
const arr1 = ["zedan", "Braa"];
const arr2 = ["Emil", "MoHamed", "Ahmid"];
const arr3 = ["Hamza", "Ali"];
const Habib_myChildren = arr1.concat(arr2, arr3);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

أيضاً أن تأخذ نصوص كوسيطات `concat()` يمكن للطريقة

مثال ↓↓ (دمج مصفوفة مع القيم) ✓

```
const arr1 = ["Emil", "MoHamed", "Ahmid"];
const Habib_myChildren = arr1.concat("Habib");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## اضافة عناصر في وسط المصفوفة

`splice()`، تضيف الطريقة عناصر جديدة إلى المصفوفة

المصفوفة للاضافة `slice()` تقطع الطريقة

## كيف اضافة عناصر في وسط المصفوفة

الطريقة لإضافة عناصر جديدة إلى المصفوفة `splice()` يمكن استخدام هذه

مثال ↓↓ ✓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];
HaBiB.splice(2, 0, "Osama", "Hamza");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تحدد المعلمة الأولى (2) الموضع الذي يجب إضافة عناصر جديدة (مقسمة إليها)

تحدد المعلمة الثانية (0) عدد العناصر التي يجب إزالتها

العناصر الجديدة التي سيتم إضافتها ("Osama" و "Hamza") تحدد بقية البرمترات

بإرجاع مصفوفة تحتوي على العناصر المحذوفة `splice()` تقوم الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.splice(2, 2, "Osama", "Hamza");
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ازالة عناصر من محددة في المصفوفة

للإزالة العناصر دون ترك "ثقوب" في المصفوفة (`splice()`) باستخدام إعدادات البرموتات الذكي، يمكنك استخدامه

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.splice(0, 1);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تحدد المعلمة الأولى (0) الموضع الذي يجب إضافة عناصر جديدة (مقسمة إليها)

. تحدد المعلمة الثانية (1) عدد العناصر التي يجب إزالتها

. يتم حذف بقية البرموتات. لن يتم إضافة أي عناصر جديدة

## كيف تجزئة المصفوفة

.بتقسيم قطعة من المصفوفة إلى مصفوفة جديدة (`slice()`) تقوم الطريقة

:يقطع هذا ال ✓ مثال ↓↓ جزءًا من مصفوفة بدءًا من عنصر المصفوفة 1 ("البرتقالي")

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Osama", "Abu Habib", "Abu Zedan"];  
const citrus = HaBiB.slice(1);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## ملحوظة

بإنشاء مصفوفة جديدة `slice()` تقوم الطريقة

بإزالة أي عناصر من المصفوفة المصدر `slice()` لا تقوم الطريقة

("Abu Habib") يقسم هذا ال `slice()` مثال `slice()` جزءًا من مصفوفة بدءًا من عنصر المصفوفة 3

✓ مثال `slice()`

```
const HaBiB= ["Al-Husini", "Osman", "Osama", "Abu Habib", "Abu Zedan"];  
const citrus = HaBiB.slice(3);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

`slice(1, 3)` وسيطتين مثل `slice()` يمكن أن تأخذ الطريقة

تقوم الطريقة بعد ذلك بتحديد العناصر من وسيطة البداية وحتى (ولكن ليس بما في ذلك) وسيطة النهاية

✓ مثال `slice()`

```
const HaBiB= ["Al-Husini", "Osman", "Osama", "Abu Habib", "Abu Zedan"];  
const citrus = HaBiB.slice(1, 3);
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

تقوم بتقسيم بقية المصفوفة `slice()` إذا تم حذف وسيطة النهاية، كما هو الحال في الأمثلة الأولى، فإن الطريقة

✓ مثال `slice()`

```
const HaBiB= ["Al-Husini", "Osman", "Osama", "Abu Habib", "Abu Zedan"];  
const citrus = HaBiB.slice(2);
```



كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف تحويل المصفوف الى نص عادي

تقوم جافاسكربت تلقائيًا بتحويل المصفوفة إلى نص مفصولة بفواصل عندما تكون القيمة الأولية متوقعة.


هذا هو الحال دائمًا عندما تحاول إخراج مصفوفة

:سيؤدي هذان ال مثال  مثال  ان إلى نفس النتيجة

 مثال 

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
document.getElementById("Habib").innerHTML = HaBiB.toString();
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

 مثال 

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
document.getElementById("Habib").innerHTML = HaBiB;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ملحوظة

toString() تحتوي جميع كائنات جافاسكربت على طريقة

## كيف إيجاد القيم القصوى والصغرى في المصفوفة

لا توجد وظائف مضمنة للعثور على أعلى أو أدنى قيمة في مصفوفة جافاسكربت

سوف تتعلم كيفية حل هذه المشكلة في الفصل التالي من هذا الكتاب

## كيف الفرز والترتيب للمصفوفات



## اشهر طرق الفرز للمصفوفة

بفرز مصفوفة أبجدياً `sort()` تقوم الطريقة

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.sort();
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف عكس ترتيب المصفوفة

بعكس العناصر الموجودة في المصفوفة `reverse()` تقوم الطريقة

:يمكنك استخدامه لفرز مصفوفة بترتيب تنازلي

✓ مثال ↓↓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
HaBiB.sort();  
HaBiB.reverse();
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف عمل ترتيب رقمي للمصفوفة

. الدالة بفرز القيم كنصوص `sort()` بشكل افتراضي، تقوم

"Al-Husini" تأتي قبل "Abu Habib" يعمل هذا بشكل جيد مع النصوص

". ومع ذلك، إذا تم فرز الأرقام كنصوص، فإن "25" أكبر من "100"، لأن "2" أكبر من "1"

.ستنتج الطريقة نتيجة غير صحيحة عند فرز الأرقام `sort()`، ولهذا السبب

: يمكنك إصلاح ذلك من خلال توفير وظيفة المقارنة

✓ مثال ↓↓

```
const hAbIB = [40, 100, 1, 5, 25, 10];  
hAbIB.sort(function(a, b){return a - b});
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

:استخدم نفس الخدعة لفرز مصفوفة تنازليًا

✓ مثال ↓↓

```
const hAbIB = [40, 100, 1, 5, 25, 10];  
hAbIB.sort(function(a, b){return b - a});
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## وظيفة المقارنة

.الغرض من وظيفة المقارنة هو تحديد ترتيب فرز بديل

:يجب أن تُرجع دالة المقارنة قيمة سالبة أو صفرية أو موجبة، اعتمادًا على الوسائط

```
function(a, b){return a - b}
```

تقارن الدالة قيمتين، فإنها ترسل القيم إلى دالة المقارنة، وتقوم بفرز القيم وفقًا للقيمة التي تم إرجاعها (سلبية، صفر، إيجابية)

**b**. يتم فرزها قبل **a**، إذا كانت النتيجة سلبية

**a**. يتم فرزها قبل **b**، إذا كانت النتيجة إيجابية

.إذا كانت النتيجة 0، فلن يتم إجراء أي تغييرات على ترتيب فرز القيمتين

✓ مثال ↓↓:

(**a, b**). تقوم دالة المقارنة بمقارنة جميع القيم الموجودة في المصفوفة، قيمتين في المرة الواحدة

.تستدعي الطريقة دالة المقارنة (**100, 40**) **sort()**، عند مقارنة 40 و100

.وبما أن النتيجة سالبة (-60)، فستقوم دالة الفرز بفرز 40 كقيمة أقل من 100. (**a - b**) تحسب الدالة 40 - 100

:يمكنك استخدام مقتطف الشفرة هذا لتجربة الفرز رقميًا وأبديًا

```
<button onclick="Husini1()">Sort Alphabetically</button>  
<button onclick="Husini2()">Sort Numerically</button>
```

```
<p id="Habib"></p>
```

```
<script>
```

```
const hAbIB = [40, 100, 1, 5, 25, 10];
```

```
document.getElementById("Habib").innerHTML = hAbIB;
```

```
function Husini1() {
```

```
  hAbIB.sort();
```

```
  document.getElementById("Habib").innerHTML = hAbIB;
```

```
}
```

```
function Husini2() {
```

```
  hAbIB.sort(function(a, b){return a - b});
```

```
  document.getElementById("Habib").innerHTML = hAbIB;
```

```
}
```

```
</script>
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## فرز مصفوفة بترتيب عشوائي

✓ مثال ↓↓

```
const hAbIB = [40, 100, 1, 5, 25, 10];
```

```
hAbIB.sort(function(){return 0.5 - Math.random()});
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## طريقة فيشر ييتس

ليس دقيقاً، وسوف تفضل بعض الأرقام على غيرها، `array.sort()`، مثال ✓ ↓↓ أعلاه

!الطريقة الصحيحة الأكثر شيوعاً، تسمى طريقة فيشر ييتس، وتم تقديمها في علم البيانات في وقت مبكر من عام 1938

:في جافاسكربت يمكن ترجمة الطريقة إلى هذا

✓ مثال ↓↓

```
const hAbIB = [40, 100, 1, 5, 25, 10];
```

```
for (let i = hAbIB.length - 1; i > 0; i--) {  
  let j = Math.floor(Math.random() * (i+1));  
  let k = hAbIB[i];  
  hAbIB[i] = hAbIB[j];  
  hAbIB[j] = k;  
}
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## ابحث عن أعلى أو أدنى قيمة للمصفوفة

لا توجد وظائف مضمنة للعثور على القيمة القصوى أو الدنيا في المصفوفة

ومع ذلك، بعد فرز مصفوفة ، يمكنك استخدام الفهرس للحصول على القيم الأعلى والأدنى

الترتيب تصاعدياً:

✓ مثال ↓↓

```
const hAbIB = [40, 100, 1, 5, 25, 10];  
hAbIB.sort(function(a, b){return a - b});  
// now hAbIB[0] contains the lowest value  
// and hAbIB[hAbIB.length-1] contains the highest value
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

الترتيب تنازلياً:

✓ مثال ↓↓

```
const hAbIB = [40, 100, 1, 5, 25, 10];  
hAbIB.sort(function(a, b){return b - a});  
// now hAbIB[0] contains the highest value  
// and hAbIB[hAbIB.length-1] contains the lowest value
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

يعد فرز مصفوفة كاملة طريقة غير فعالة للغاية إذا كنت تريد فقط العثور على القيمة الأعلى (أو الأدنى)

## Math.max() استخدام

للعثور على أعلى رقم في مصفوفة **Math.max.apply** يمكنك استخدامه

✓ مثال ↓↓

```
function Habib_ArrayMax(arr) {  
  return Math.max.apply(null, arr);  
}
```

كتاب الكافي في جافاسكريبت الجزء الأول ابو حبيب الحسيني

**Math.max(1, 2, 3)** يعادل **Math.max.apply(null, [1, 2, 3])**.

## Math.min() استخدام

للعثور على أقل رقم في المصفوفة **Math.min.apply** يمكنك استخدامه

✓ مثال ↓↓

```
function Habib_ArrayMin(arr) {  
  return Math.min.apply(null, arr);  
}
```

كتاب الكافي في جافاسكريبت الجزء الأول ابو حبيب الحسيني

**Math.min(1, 2, 3)** يعادل **Math.min.apply(null, [1, 2, 3])**.

## Min / Max طرق خاصة بي

"الحل الأسرع هو استخدام طريقة "شائعة الاستخدام وسهلة

:تتكرر هذه الوظيفة عبر مصفوفة تقارن كل قيمة بأعلى قيمة تم العثور عليها

مثال ↓↓ (البحث عن الحد الأقصى) ✓

```
function Habib_ArrayMax(arr) {  
  let len = arr.length;  
  let max = -Infinity;  
  while (len--) {  
    if (arr[len] > max) {  
      max = arr[len];  
    }  
  }  
  return max;  
}
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

تتكرر هذه الوظيفة عبر مصفوفة تقارن كل قيمة بأقل قيمة تم العثور عليها

مثال ↓↓ (البحث عن الحد الأدنى) ✓

```
function Habib_ArrayMin(arr) {  
  let len = arr.length;  
  let min = Infinity;  
  while (len--) {  
    if (arr[len] < min) {  
      min = arr[len];  
    }  
  }  
  return min;  
}
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## فرز مصفوفات الكائنات

تحتوي مصفوفات جافاسكربت غالبًا على كائنات

✓ مثال ↓↓

```
const HaBiB= [  
  {use:"Hamza", year:2016},  
  {use:"Habib", year:2001},  
  {use:"Abu Habib Al-Husini", year:2010}  
];
```

فيمكن استخدام الطريقة لفرز المصفوفة `sort()` حتى لو كانت الكائنات لها خصائص أنواع بيانات مختلفة:  
الحل هو كتابة دالة مقترنة لمقارنة قيم الخاصية

✓ مثال ↓↓

```
HaBiB.sort(function(a, b){return a.year - b.year});
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

:تعد مقترنة خصائص النص أكثر تعقيداً بعض الشيء

✓ مثال ↓↓

```
HaBiB.sort(function(a, b){  
  let x = a.type.toLowerCase();  
  let y = b.type.toLowerCase();  
  if (x < y) {return -1;}  
  if (x > y) {return 1;}  
  return 0;  
});
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## تكرار مصفوفة جافاسكربت

تعمل طرق تكرار المصفوفة على كل عنصر من عناصر المصفوفة

## forEach ()

دالة (دالة رد اتصال) مرة واحدة لكل عنصر من عناصر المصفوفة `forEach()` تستدعي الطريقة

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let hAbiB = "";  
Habib_Num.forEach(Husini);  
  
function Husini(value, index, array) {  
  hAbiB += value + "<br>";  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

يستخدم ال ✓ مثال ↓↓ أعلاه معلمة القيمة فقط. يمكن إعادة كتابة ال ✓ مثال ↓↓ إلى

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let hAbiB = "";  
Habib_Num.forEach(Husini);  
  
function Husini(value) {  
  hAbiB += value + "<br>";  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## استخدام دالة الخريطة

بإنشاء مصفوفة جديدة عن طريق تنفيذ دالة على كل عنصر من عناصر المصفوفة (**map()**) تقوم الطريقة

بتنفيذ الوظيفة لعناصر المصفوفة بدون قيم (**map()**) لا تقوم الطريقة

• لا تغير الطريقة المصفوفة الأصلية (**map()**)

يقوم هذا ال ✓ مثال ↓↓ بضرب كل قيمة مصفوفة بمقدار 2



✓ مثال ↓↓

```
const Habib_Num1 = [45, 4, 9, 16, 25];  
const Habib_Num2 = Habib_Num1.map(Husini);
```

```
function Husini(value, index, array) {  
  return value * 2;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

: عندما تستخدم دالة رد الاتصال معلمة القيمة فقط، يمكن حذف معلمات الفهرس والمصفوفة

✓ مثال ↓↓

```
const Habib_Num1 = [45, 4, 9, 16, 25];  
const Habib_Num2 = Habib_Num1.map(Husini);
```

```
function Husini(value) {  
  return value * 2;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف تصفية المصفوفة والبحث فيها

بإنشاء مصفوفة جديدة تحتوي على عناصر المصفوفة التي تجتاز الاختبار (**filter()**) تقوم الطريقة

:ينشئ هذا ال ✓ مثال ↓↓ مصفوفة جديدة من عناصر ذات قيمة أكبر من 18

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
const over18 = Habib_Num.filter(Husini);
```

```
function Husini(value, index, array) {
```

```
return value > 18;
```

```
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

في ال ✓ مثال ↓↓ أعلاه، لا تستخدم وظيفة رد الاتصال معلمات الفهرس والمصفوفة، لذا يمكن حذفها

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
const over18 = Habib_Num.filter(Husini);
```

```
function Husini(value) {  
  return value > 18;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## كيف التقليل والتصفية للمصفوفات

بتشغيل دالة على كل عنصر من عناصر المصفوفة لإنتاج (تقليلها إلى) قيمة واحدة (`reduce()`) تقوم الطريقة

`reduceRight()` من اليسار إلى اليمين في المصفوفة. أنظر أيضا `reduce()` تعمل الطريقة

. لا تقلل الطريقة من المصفوفة الأصلي (`reduce()`)

: يجد هذا ال ✓ مثال ↓↓ مجموع جميع الأرقام في مصفوفة

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let sum = Habib_Num.reduce(Husini);
```

```
function Husini(total, value, index, array) {  
  return total + value;  
}
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 4 وسيطات

- الإجمالي (القيمة الأولية / القيمة التي تم إرجاعها مسبقاً)
- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

لا يستخدم ال ✓ مثال ↓↓ أعلاه معلمات الفهرس والمصفوفة . يمكن إعادة كتابته إلى

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let sum = Habib_Num.reduce(Husini);  
  
function Husini(total, value) {  
  return total + value;  
}
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

قيمة أولية `reduce()` يمكن أن تقبل الطريقة

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let sum = Habib_Num.reduce(Husini, 100);  
  
function Husini(total, value) {  
  return total + value;  
}
```

## كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

### ازاحة المصفوفة الى اليمين

بتشغيل دالة على كل عنصر من عناصر المصفوفة لإنتاج (تقليها إلى) قيمة واحدة `reduceRight()` تقوم الطريقة

`reduce()` من اليمين إلى اليسار في المصفوفة. أنظر أيضا `reduceRight()` الأعمال

لا تقلل الطريقة من المصفوفة الأصلي `reduceRight()`

: يجد هذا ال ✓ مثال ↓↓ مجموع جميع الأرقام في مصفوفة

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let sum = Habib_Num.reduceRight(Husini);  
  
function Husini(total, value, index, array) {  
  return total + value;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 4 وسيطات

- الإجمالي (القيمة الأولية / القيمة التي تم إرجاعها مسبقاً).
- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

لا يستخدم ال ✓ مثال ↓↓ أعلاه معلمات الفهرس والمصفوفة . يمكن إعادة كتابته إلى

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let sum = Habib_Num.reduceRight(Husini);  
  
function Husini(total, value) {  
  return total + value;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## تابع طرق المصفوفة

مما إذا كانت جميع قيم المصفوفة تجتاز الاختبار **every()** تتحقق الطريقة

يتحقق هذا ال ✓ مثال ↓↓ مما إذا كانت جميع قيم المصفوفة أكبر من 18

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 45, 4, 9, 16, 45, 4, 9, 16, 25];  
let All_Habib_Num_Over18 = Habib_Num.every(Husini);  
  
function Husini(value, index, array) {
```

```
return value > 18;
```

```
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

عندما تستخدم دالة رد الاتصال المعلمة الأولى فقط (القيمة)، فيمكن حذف البرمات الأخرى

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let All_Habib_Num_Over18 = Habib_Num.every(Husini);
```

```
function Husini(value) {  
  return value > 18;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## طريقة أخرى للبحث

مما إذا كانت بعض قيم المصفوفة تجتاز الاختبار `some()` تتحقق الطريقة

يتحقق هذا ال ✓ مثال ↓↓ مما إذا كانت بعض قيم المصفوفة أكبر من 18

✓ مثال ↓↓

```
const Habib_Num = [45, 4, 9, 16, 25];  
let someOver18 = Habib_Num.some(Husini);
```

```
function Husini(value, index, array) {  
  return value > 18;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة السلعة

- فهرس البند
- المصفوفة نفسها

## كيف التحكم عن طريق الفهارس

في مصفوفة عن قيمة عنصر وترجع موضعها (`indexOf()`) تبحث الطريقة

ملاحظة: العنصر الأول له الموضع 0، والعنصر الثاني له الموضع 1، وهكذا

مثال ↓↓ ✓

"Abu Habib": ابحث في مصفوفة عن العنصر

```
const HaBiB= ["Abu Habib", "Osman", "Abu Habib", "Abu Zedan"];  
let position = HaBiB.indexOf("Abu Habib") + 1;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

بناء الجملة

`array.indexOf(item, start)`

**item** مطلوب. العنصر المطلوب البحث عنه.  
اختياري. من أين تبدأ البحث. ستبدأ القيم السالبة عند الموضع المحدد، ويتم العد من النهاية، والبحث حتى النهاية.  
**start**

ترجع -1 إذا لم يتم العثور على العنصر (`Array.indexOf()`)

إذا كان العنصر موجوداً أكثر من مرة، فإنه يُرجع موضع التواجد الأول

## ( ) lastIndexOf مصفوفة جافاسكريبت

ولكنه يُرجع موضع آخر تواجد للعنصر المحدد، (`Array.indexOf()`) هو نفسه (`Array.lastIndexOf()`)

## مثال ↓↓ ✓

"Abu Habib" يبحث في مصفوفة عن العنصر

```
const HaBiB= ["Abu Habib", "Osman", "Abu Habib", "Abu Zedan"];  
let position = HaBiB.lastIndexOf("Abu Habib") + 1;
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## بناء الجملة

`array.lastIndexOf(item, start)`

<code>item</code>	مطلوب. العنصر المطلوب البحث عنه
<code>start</code>	خيارى. من أين تبدأ البحث. ستبدأ القيم السالبة عند الموضع المحدد من النهاية، وتبحث حتى البداية

## طريقة اخرى للبحث

قيمة عنصر المصفوفة الأول الذي يمرر دالة اختبار `find()` تُرجع الطريقة

يبحث هذا ال ✓ مثال ↓↓ عن (يعيد قيمة) العنصر الأول الأكبر من 18

## مثال ↓↓ ✓

```
const Habib_Num = [4, 9, 16, 25, 29];  
let first = Habib_Num.find(Husini);  
  
function Husini(value, index, array) {  
  return value > 18;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

## دعم المتصفح

**find()** (جافاسكربت 2015 ES6) هي إحدى ميزات **find()**.

نوهو مدعوم في جميع المتصفحات الحديثة

**find()** غير مدعوم في Internet Explorer.

## دالة () findIndex

بإرجاع فهرس المصفوفة الأول الذي يمرر وظيفة الاختبار **findIndex()** تقوم الطريقة

يعثر هذا ال ✓ مثال ↓↓ على فهرس العنصر الأول الأكبر من 18

✓ مثال ↓↓

```
const Habib_Num = [4, 9, 16, 25, 29];  
let first = Habib_Num.findIndex(Husini);
```

```
function Husini(value, index, array) {  
  return value > 18;  
}
```

كتاب الكافي في جافاسكربت الجزء الأول ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة السلعة
- فهرس البند
- المصفوفة نفسها

## دعم المتصفح

**findIndex()** (جافاسكربت 2015 ES6) هي إحدى ميزات **findIndex()**.

نوهو مدعوم في جميع المتصفحات الحديثة

**findIndex()** غير مدعوم في Internet Explorer.



## Array.from()

يلرجع كائن مصفوفة من أي كائن له خاصية الطول أو أي كائن قابل للتكرار **Array.from()** تقوم الطريقة

✓ مثال ↓↓

: إنشاء مصفوفة من نص

```
Array.from("ABCDEFGG");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## دعم المتصفح

**from()** (جافاسكربت 2015 ES6) هي إحدى ميزات **from()**.

نوهو مدعوم في جميع المتصفحات الحديثة

**from()** غير مدعوم في Internet Explorer.

## كيف الحصول على مفاتيح المصفوفة

مع مفاتيح المصفوفة **Array Iterator** يلرجع كائن **Array.keys()** تقوم الطريقة

✓ مثال ↓↓

:الذي يحتوي على مفاتيح المصفوفة. **Array Iterator** قم بإنشاء كائن

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
const keys = HaBiB.keys();
```

```
for (let x of keys) {  
  text += x + "<br>";  
}
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## دعم المتصفح

(جافاسكريبت 2015 ES6) هي إحدى ميزات (**keys()**).

نوهو مدعوم في جميع المتصفحات الحديثة

Internet Explorer غير مدعوم في (**keys()**).

## إدخالات المصفوفة

✓ مثال ↓↓

:أنشئ مكرر المصفوفة، ثم كرره على أزواج المفتاح/القيمة

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
const f = HaBiB.entries();
```

```
for (let x of f) {  
  document.getElementById("Habib").innerHTML += x;  
}
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

:مع أزواج المفتاح/القيمة Array Iterator بإرجاع كائن (**entries()**) تقوم الطريقة

```
[0. "موز"]  
[1. "برتقالي"]  
[2. "تفاحة"]  
[3. "مانجو"]
```

. لا تغير الطريقة المصفوفة الأصلية (**entries()**).

## دعم المتصفح

(جافاسكريبت 2015 ES6) هي إحدى ميزات (**entries()**).

نوهو مدعوم في جميع المتصفحات الحديثة

Internet Explorer غير مدعوم في (**entries()**).

## التضمن في المصفوفة

إلى المصفوفات. يتيح لنا هذا التحقق من وجود عنصر في المصفوفة (**Array.includes()** ECMAScript 2016 تم تقديم **Indexof** على عكس **NaN**، بما في ذلك).

مثال ↓↓ ✓

```
const HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];
```

```
HaBiB.includes("Abu Zedan");// is true
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسينى

### بناء الجملة

**array.includes(search-item)**

**Array.indexof()** على عكس **NaN**. بالتحقق من قيم **Array.includes()** يسمح

**Array.includes()** و **Edge 12/13** غير مدعوم في و

إصدارات المتصفح الأولى مع الدعم الكامل هي:

## دعم المتصفح

**ECMAScript 2016** هي إحدى ميزات **includes()**

وهو مدعوم في جميع المتصفحات الحديثة

## استخدام **const** مع المصفوفة

### (ES6) **إيكماسكريبت 2015**

**const**: في عام 2015، قدمت جافاسكريبت كلمة محجوزة جديدة مهمة

**const:** لقد أصبح من الممارسات الشائعة الإعلان عن المصفوفات باستخدام

✓ مثال ↓↓

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## لا يمكن إعادة تعيينها

**const:** لا يمكن إعادة تعيين مصفوفة تم الإعلان عنه

✓ مثال ↓↓

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];  
HaBiB= ["Abu Bakr Al-Siddiq!!!", "Hamza", "Omar ibn Al-khattab"]; // ERROR
```

كتاب الكافي في جافاسكريبت الجزء الاول ابو حبيب الحسيني

## المصفوفات ليست ثوابت

مضللة بعض الشيء، **const** الكلمة المحجوزة

. لا يحدد مصفوفة ثابتاً. يحدد مرجعاً ثابتاً إلى مصفوفة

.ولهذا السبب، لا يزال بإمكاننا تغيير عناصر المصفوفة الثابتة

## يمكن إعادة تعيين العناصر

:يمكنك تغيير عناصر مصفوفة ثابتة

✓ مثال ↓↓

```
// You can create a constant array:  
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

```
// You can change an element:  
HaBiB[0] = "Abu Bakr Al-Siddiq!!!";
```

// You can add an element:

```
HaBiB.push("Omar ibn Al-khattab");
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## دعم المتصفح

المحجوزة غير مدعومة في 10 أو إصدار سابق **const** الكلمة

:المحجوزة **const** يحدد الجدول التالي إصدارات المتصفح الأولى مع الدعم الكامل للكلمة

## تم تعيينه عندما أعلن

:يجب تعيين قيمة لمتغيرات جافاسكربت عند الإعلان عنها **const**

.المعنى: يجب تهيئة المصفوفة التي تم الإعلان عنها عند الإعلان عنها **const**

:دون تهيئة المصفوفة خطأ في بناء الجملة **const** يعد الاستخدام

✓ مثال ↓↓

:هذا لن يعمل

```
const HaBiB;
```

```
HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

.يمكن تهيئة المصفوفات المعلنة في أي وقت **var**

:يمكنك أيضاً استخدام المصفوفة قبل الإعلان عنها

✓ مثال ↓↓

:لابأس

```
HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];
```

```
var HaBiB;
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

## نطاق كتلة كونست

Block Scope تحتوي على **const** المصفوفة التي تم الإعلان عنها

:المصفوفة المعلن عنها في كتلة ليست هي نفسها المصفوفة المعلن عنها خارج الكتلة

✓ مثال ↓↓

```
const HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];  
// Here HaBiB[0] is "Habib"  
{  
  const HaBiB= ["Abu Bakr Al-Siddiq!!!", "Hamza", "Abu Habib Al-Husini"];  
  // Here HaBiB[0] is "Abu Bakr Al-Siddiq!!!"  
}  
// Here HaBiB[0] is "Habib"
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

:ب لا تحتوي على نطاق كتلة **var** المصفوفة التي تم الإعلان عنها

✓ مثال ↓↓

```
var HaBiB= ["Habib", "Hamza", "Abu Habib Al-Husini"];  
// Here HaBiB[0] is "Habib"  
{  
  var HaBiB= ["Abu Bakr Al-Siddiq!!!", "Hamza", "Abu Habib Al-Husini"];  
  // Here HaBiB[0] is "Abu Bakr Al-Siddiq!!!"  
}  
// Here HaBiB[0] is "Abu Bakr Al-Siddiq!!!"
```

كتاب الكافي في جافاسكربت الجزء الاول ابو حبيب الحسيني

للمزيد اراء كتاب المرجع الكامل لجافا سكربت

## إعادة تعريف المصفوفات

:في أي مكان في البرنامج **var** يُسمح بإعادة الإعلان عن مصفوفة تم الإعلان عنها

## ✓ مثال ↓↓

```
var HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
var HaBiB= ["Abu Bakr Al-Siddiq!!!", "Abu Habib Al-Husini"]; // Allowed
HaBiB= ["Hamza", "Habib"]; // Allowed
```

إلى نفس النطاق أو في نفس الكتلة **const** لا يُسمح بإعادة الإعلان عن مصفوفة أو إعادة تعيينها

## ✓ مثال ↓↓

```
var HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed
{
  var HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
  const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed
}
```

في نفس النطاق، أو في نفس الكتلة، **const** لا يُسمح بإعادة الإعلان عن مصفوفة موجودة أو إعادة تعيينها

## ✓ مثال ↓↓

```
const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed
var HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed
HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed

{
  const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
  const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed
  var HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed
  HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Not allowed
}
```

يُسمح بإعادة تعريف مصفوفة بـ ، في نطاق آخر، أو في كتلة أخرى **const**

## ✓ مثال ↓↓

```
const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
{
  const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
}
{
```

```
const HaBiB= ["Hamza", "Abu Habib Al-Husini"]; // Allowed
```

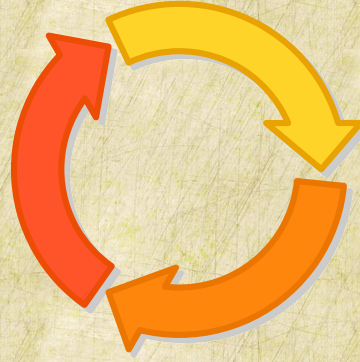
```
}
```

## كتاب المرجع الكامل لجافاسكريبت

للحصول على مرجع كامل لجميع الدوال في جافا سكريبت حمل مرجع جافاسكريبت من موقع القافلة المهنية

أبو حبيب الحسيني





ابو حبيب الحارثي

# التكملة في الجزء الثاني

## باذن الله تعالى