

الكافى

فى

JavaScript

الجزء الثالث

أبو حبيب الحسينى

ابو حبيب الحسينى

المرجع العربي السريع وردبريس



ابو حبيب الحسينى

بإذن الله تعالى سيتم نشر أكبر مرجع عربى سريع للوردبريس
يشرح 18 الف داله واجراء باللغة العربية للوردبريس انشاء وترجمة
الذكاء الاصطناعى والترجمة جيدة الى حد ما ومعنى مرجع سريع
اي شرح سريع بدون امثلة وسيذكر الكتاب اهمية كلاسات
ومكتبات الوردبريس التى يغفل عنها الكثير و هى شبيها بالذكاء

الاصتناعى فى جلب البيانات من اى مكان بشتى الطرق
والوسائل والتحكم فى الويب بصورة جنونية وتحميل مواقع
كاملة وارسال الاف الايميلات واشياء اخرى لن تتخيلها ولا يعرفها
الكثير عن قوة هذه المكتبات بمجرد تنصيب سيرفر اباتشى
على جهازك ستستطيع استخدام مكتبات ودوال الوردبريس
بمنتهى السهولة بعد تضمناها والتي سنصدر لها سلسلة كاملة
مدعومة بالامثلة ان شاء الله تعالى

بو حبيب الحسينى

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

مثل على هذا الكلام

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

لاحظ هنا ان الجملة اصبحت غير مفهومة او غير مرتبة لان بعض الكلمات نقلت من مكانها بسبب وضع كلمات انجليزية وسط الجمل العربية وهذا عيب فى الترميز يو تى اف ففى مثل هذه الحالات حاول ان تستنتج الجمله بنفسك وتفهما

حاولنا تقليل هذا العيب قدر المستطاع باستخدام بعض المصطلحات الانجليزية باللغة العربية مثل (سي اس اس) و (نود جي اس) وهكذا لنقل من هذا قدر المستطاع

أبو حبيب الحسيني

فهرس الكتاب

ملحوظة مهمة جدا.....	5
فهرس الكتاب.....	7
أخطاء جافاسكربت الشائعة.....	19
احذر هذا الخطا.....	19
انواع المقارنة فى الوضع الشرطى.....	20
احظر: من طرق الاضافة فى النتيجة.....	21
سوء الفهم للالة.....	23
مثال.....	23
كيف تقسيم الكود.....	23
مثال 1.....	23
مثال 2.....	24
مثال 3.....	24
وضع فاصلة منقوطة فى غير موضعها.....	24
اجراء عمليات بكلمة ريتورن من الاجراء.....	25
مثال 1.....	25
مثال 2.....	25
مثال 3.....	26
مثال 4.....	26
مثال 5.....	27
توضيح.....	27
جعل مفاتيح المصفوفة نصوص.....	28
مثال.....	28
مثال:.....	29
إنهاء البيانات بفاصلة.....	29
مثال الكائن:.....	29
مثال المصفوفة:.....	29
جسون:.....	30
جسون:.....	30
القيمة الغير محددة و الغير فارغة.....	30
مثال:.....	31
غير صحيح:.....	31
غير صحيح:.....	31
صحيح:.....	31

أدوات.....	32
تقليل النشاط في الحلقات.....	32
سيء:.....	32
كود أفضل:.....	32
تقليل الوصول إلى DOM.....	33
مثال.....	33
تقليل حجم DOM.....	33
تجنب المتغيرات غير الضرورية.....	33
تأخير تحميل.....	34
مثال.....	34
تجنب الاستخدام كلمة with.....	35
احذر تمت إزالة هذه الكلمات المحجوزة.....	36
أشهر الكلمات المحجوزة المستخدمة.....	37
الكلمات المحجوزة.....	37
كلمات محجوزة أخرى.....	37
معالجات أحداث HTML.....	39
الاستخدام الصارم.....	39
الوصول إلى كل حرف في النصوص.....	40
مثال.....	40
مثال.....	40
كتابة الكود على سطور متعددة.....	41
مثال.....	41
مثال.....	42
استخدام الكلمات المحجوزة كأسماء الخصائص.....	42
مثال الكائن.....	42
إزالة المسافات من الجوانب.....	42
مثال.....	42
دالة isArray().....	43
مثال.....	43
جلب كل العناصر.....	43
مثال.....	43
تنفيذ إجراء على كل العناصر بدالة الخريطة.....	44
مثال.....	44
الفلتر.....	44
مثال.....	45

reduce دالة.....	45
مثال.....	45
reduceRight دالة.....	46
مثال.....	46
داله اخرى للعمليات على المصفوفة	46
مثال.....	46
some دالة.....	47
مثال.....	47
IndexOf دالة.....	47
مثال.....	48
lastIndexOf () دالة.....	48
مثال.....	48
JSON.parse().....	48
JSON.stringify().....	49
مثال.....	50
كيف جلب التاريخ.الآن	50
مثال.....	50
ISOString() تحويل التاريخ إلى	51
مثال.....	51
JSON() تحويل التاريخ إلى	51
مثال.....	51
شاهد هذه الامثلة	52
مثال.....	52
مثال.....	52
مثال.....	53
Object.defineProperty().....	54
مثال.....	54
مثال.....	55
مثال.....	56
ربط الدالة.....	57
مثال.....	57
فواصل زائدة.....	58
مثال الكائن.....	58
مثال المصفوفة.....	58
كائنات جسون:.....	59

JSON: مصفوقات.....	59
اعادة الاعلان.....	60
مثال.....	60
الثابت فى.....	60
مثال.....	60
وظائف مختصرة.....	61
مثال.....	61
مثال.....	62
عامل الانتشار.....	62
مثال.....	62
مثال.....	63
حلقة For/Of.....	63
التكرار فوق مصفوفة.....	64
مثال.....	64
التكرار على نص.....	64
مثال.....	64
الاعلان عن كائن الخريطة.....	65
مثال.....	65
مجموعات.....	65
مثال.....	65
فئات.....	66
بناء الجملة.....	66
مثال.....	66
باستخدام الكلاسات.....	67
مثال.....	67
كائن الوعود فى.....	67
انشاء الجملة.....	68
Promise() مثال باستخدام ال.....	68
نوع الاجراء او الكود.....	69
مثال.....	69
ملحوظة.....	70
قيم البرمترات الافتراضية.....	70
مثال.....	70
البرامتر غير محدود الاعداد.....	71
مثال.....	71

دالة تتضمن .	71
مثال	71
دالة startsWith	72
مثال	72
String.endsWith()	72
مثال	72
.Array.from()	73
مثال	73
جلب المفاتيح	73
مثال	73
طرق البحث	74
مثال	74
الدالة ()findIndex	74
مثال	75
طرق الرياضيات	75
طريقة Math.trunc()	76
مثال	76
طريقة Math.sign()	76
مثال	76
طريقة Math.cbrt()	77
مثال	77
طريقة Math.log2()	77
مثال	77
طريقة Math.log10()	77
مثال	77
خصائص كلاس الارقام	78
مثال إبسيلون	78
مثال MIN_SAFE_INTEGER	78
مثال MAX_SAFE_INTEGER	78
طرق الأرقام	79
طريقة Number.isInteger()	79
مثال	79
طريقة Number.isSafeInteger()	79
مثال	80
أساليب عامة جديدة	80

isFinite() طريقة.....	80
مثال.....	81
isNaN() طريقة.....	81
مثال.....	81
إلادخالات والتكرار.....	81
مثال.....	81
وحدات.....	82
الاستيراد من التصديرات النصية.....	82
الاستيراد من التصديرات الافتراضية.....	83
ايكما 2016 ميزات جديدة في.....	83
عامل الأس.....	83
مثال.....	84
مثال.....	84
مهمة الأس.....	84
مثال.....	84
تابع التضمن للمصفوفة.....	85
مثال.....	85
نص الربط.....	85
أمثلة.....	86
إدخالات الكائن.....	86
مثال.....	86
مثال.....	87
مثال.....	87
قيم الكائنات.....	88
مثال.....	88
وظائف غير المتزامنة.....	88
في انتظار المهلة للتنفيذ.....	88
التكرار غير المتزامن.....	89
مثال.....	89
وأخيرَ كائن ال Promise().....	90
مثال.....	90
جلب الخصائص بالطريقة الحديثة.....	90
مثال.....	90
مواضيع.....	91
الذاكرة المشتركة.....	91

SharedArrayBuffer.....	91
TrimStart () دالة.....	91
مثال.....	92
TrimEnd () دالة.....	92
مثال.....	92
fromEntries() كائن.....	93
مثال.....	93
كشف الاخطاء.....	93
مثال.....	93
flat() دالة.....	94
مثال.....	94
flatMap () الدالة.....	95
مثال.....	95
الفرز.....	95
مثال.....	96
JSON.stringify الوظيفة.....	97
مثال.....	97
الرموز الفاصلة.....	97
مثال.....	97
ملحوظة.....	98
String () الكلاس.....	98
مثال.....	99
matchAll () دالة.....	99
مثال.....	99
مثال.....	99
مثال.....	100
عامل الدمج الفارغ (??).....	100
مثال.....	100
مشغل التسلسل الاختياري (?.).....	100
مثال.....	101
=&& المشغل.....	101
مثال منطقي والواجب.....	101
= المشغل.....	101
مثال منطقي أو التنازل.....	102
ال =?? المشغل.....	102

.....مثال على مهمة الدمج الفارغة	102
.....البحث والاستبدال لكل التطابقات دفعة واحدة	103
.....مثال	103
.....مثال	103
.....استخدام الفاصلة (_)	103
.....مثال	104
.....مثال	104
.....مثال	104
.....ملحوظة	104
.....النهاية الماسوية للإنترنت إكسبلورر	105
.....تاريخ رفع الدعم عن انترنت اكسبلورار في 17 أغسطس	105
.....2020:	105
.....متصفح عمو غوغل	105
.....الابجيكيت	106
.....البدائيات	107
.....أمثلة للانواع	107
.....غير قابل للتغيير	107
.....الكائنات هي المتغيرات	108
.....مثال	108
.....مثال	108
.....مثال	109
.....طرق الكائن	109
.....إنشاء كائن	109
.....باستخدام كائن حرفي	110
.....مثال	110
.....مثال	110
.....مثال	111
.....باستخدام الكلمة المحجوزة الجديدة	111
.....مثال	111
.....كائنات قابلة للتغيير	112
.....مثال	112
.....خصائص الكائن	113
.....الخصائص	113
.....الوصول إلى الخصائص	113
.....مثال 1	114

مثال 2.....	114
الحلقة فور ان.....	114
بناء الجملة.....	115
مثال.....	115
إضافة خصائص جديدة.....	115
مثال.....	116
حذف الخصائص.....	116
مثال.....	116
مثال.....	117
الكائنات المتداخلة.....	117
مثال.....	117
مثال.....	118
مثال.....	118
مثال.....	118
مثال.....	119
المصفوفات والكائنات المتداخلة.....	119
مثال.....	119
مثال.....	120
سمات الخاصية.....	120
خصائص النموذج الأولي.....	121
أساليب كائن.....	121
مثال.....	121
أنظر أيضا:.....	122
الطرق.....	122
الوصول إلى أساليب الكائن.....	122
مثال.....	123
مثال.....	123
إضافة طريقة إلى كائن.....	123
مثال.....	123
استخدام الطرق المضمنة.....	124
مثال.....	124
عرض الكائنات.....	124
كيفية عرض الكائنات ؟.....	124
مثال.....	125
عرض خصائص الكائن.....	125

مثال.....	125
عرض الكائن في حلقة.....	126
مثال.....	126
استخدام Object.values().....	127
مثال.....	127
استخدام JSON.stringify().....	128
مثال.....	128
تسلسل التواريخ.....	129
مثال.....	129
Stringify وظائف.....	130
مثال.....	130
مثال.....	130
تصنيف المصفوفات.....	131
مثال.....	131
ملحقات الكائنات.....	132
(Getters وSetters) ملحقات.....	132
(الحصول على الكلمة المحجوزة) Getter.....	132
مثال.....	132
الوصول السريع للبيانات.....	133
مثال.....	133
تابع جلب البيانات.....	134
مثال 1.....	134
مثال 2.....	134
جودة الكلمات.....	135
مثال.....	135
مثال.....	136
Getters وSetters لماذا استخدام.....	137
Object.defineProperty().....	137
مثال مضاد.....	137
منشئي الكائنات.....	138
مثال.....	138
ملحوظات.....	139
أنواع الكائنات (المخططات) (الفئات).....	139
إضافة خاصية إلى كائن.....	140
مثال.....	140

إضافة طريقة إلى كائن	140
مثال	140
إضافة خاصية إلى منشئ	141
مثال	141
مثال	141
إضافة طريقة إلى منشئ	142
مثال	142
مثال	142
الآن يمكنك تجربة:	143
كائنات جاهزة فى اللغة	143
هل كنت تعلم؟	144
مثال	144
كائنات النص	145
كائنات الأرقام	145
الكائنات المنطقية	145
نماذج كائنات	146
مثال	146
مثال	147
مثال	147
وراثة النموذج الأولي	147
إضافة خصائص وأساليب للكائنات	148
باستخدام خاصية النموذج الأولي	148
مثال	148
مثال	149
عناصر التكرارية	149
التكرار على نص	150
مثال	150
التكرار على مصفوفة	150
مثال	150
استخدام نكست	151
طريقة نكست والارجاع المتعدد	151
مثال	151
مثال	152
مثال	153
مجموعات	154

تعيين الأساليب.....	154
كيفية إنشاء مجموعة.....	155
طريقة () Set.....	155
مثال.....	155
مثال.....	155
مثال.....	156
الإضافة.....	157
مثال.....	157
مثال.....	157
طريقة forEach().....	157
مثال.....	158
القيم.....	158
مثال.....	158
مثال.....	159
المفاتيح.....	159
مثال.....	159
الإدخالات.....	160
مثال.....	160
المجموعات هي كائنات.....	161
تغيير قيمة الخاصية.....	161
بناء الجملة.....	161
مثال.....	162
تغيير كلمات التعريف.....	162
قائمة جميع الخصائص.....	163
مثال.....	163
قائمة خصائص لا تعد ولا تحصى.....	164
مثال.....	164
إضافة.....	164
مثال.....	165
Getters وSetters إضافة.....	165
مثال.....	165
مثال.....	166
مثال.....	166
بيانات الوظائف.....	167
إعلانات الوظيفة.....	167

.....مثال	168
التعبيرات الوظيفية.....	168
.....مثال	168
.....مثال	169
منشئ الوظيفة.....	169
.....مثال	169
.....مثال	170
وظيفة الرفع.....	170
وظائف الاستدعاء الذاتي.....	171
.....مثال	171
استخدام الوظائف كقيم.....	172
.....مثال	172
.....مثال	172
الوظائف هي كائنات.....	172
.....مثال	173
.....مثال	173
الوظائف المختصرة.....	174
.....مثال	174
.....مثال	174
معلومات الوظيفة.....	175
معلومات الوظيفة او البرمترات.....	175
قواعد البرامتر.....	175
البرمترات الافتراضية.....	176
.....مثال	176
قيم البرمترات الافتراضية.....	176
.....مثال	176
البرامتر الغير محدود العدد.....	177
.....مثال	177
كائن البرمترات.....	177
.....مثال	178
.....مثال	178
تمرير البرامتر حسب القيمة.....	179
تمرير الكائنات حسب المرجع.....	179
استدعاء وظيفة.....	180
استدعاء وظيفة كوظيفة.....	180

مثال.....	180
ملحوظة.....	181
مثال.....	181
الكائن العام.....	182
مثال.....	182
استدعاء وظيفة كطريقة.....	182
مثال.....	183
مثال.....	183
استدعاء وظيفة مع منشئ الوظيفة.....	184
مثال.....	184
التكملة فى الجزء الرابع.....	187
باذن الله تعالى.....	187

أبو حبيب الحسينى

أخطاء جافاسكربت الشائعة

يشير هذا الفصل إلى معرفة بعض أخطاء جافاسكربت الشائعة.

احذر هذا الخطأ

قد تولد برامج جافاسكربت نتائج غير متوقعة إذا استخدم المبرمج عن طريق `if` الخطأ عامل التعيين (`=`)، بدلاً من عامل المقارنة (`==`) في عبارة

:لا تساوي `x 10` لأن (`x` كما هو متوقع) `false` تُرجع هذه العبارة `if`

```
let x = 0;
```

```
if (x == 10)
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

لأن 10 صحيح، (ربما ليس كما هو متوقع) **true** الكلمة **if** يعود هذا

```
let x = 0;  
if (x = 10)
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

لأن 0 خطأ، (ربما ليس كما هو متوقع) **false** الكلمة **if** يعود هذا

```
let x = 0;  
if (x = 0)
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

. تقوم المهمة دائماً بإرجاع القيمة

انواع المقارنة في الوضع الشرطي

:الكلمة صحيحا **if** في المقارنة العادية، لا يهم نوع الكلمات. يعود هذا

```
let x = 10;  
let y = "10";  
if (x == y)
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

:العبرة خطأ **if** في المقارنة الصارمة، نوع الكلمات مهم. تُرجع هذه

```
let x = 10;  
let y = "10";  
if (x === y)
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

العبارات تستخدم المقارنة الصلومة **switch** من الأخطاء الشائعة أن ننسى أن **case switch** سيعرض هذا تنبيهًا

```
let x = 10;  
switch(x) {  
  case 10: alert("Abo Habib Al Hosini 😊😊");  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

تنبيهًا **case switch** لن يعرض هذا

```
let x = 10;  
switch(x) {  
  case "10": alert("Abo Habib Al Hosini 😊😊");  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

احظر: من طرق الإضافة في النتيجة

. الإضافة تتعلق بإضافة أرقام

. التسلسل يدور حول إضافة نصوص

.في جافاسكربت، تستخدم كلتا العمليتين نفس +العامل

ولهذا السبب، فإن إضافة رقم كرقم سيؤدي إلى نتيجة مختلفة عن إضافة رقم
: كنص

```
| let x = 10;  
| x = 10 + 5; // Now x is 15
```

```
| let y = 10;  
| y += "5"; // Now y is "105"
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

:عند إضافة متغيرين، قد يكون من الصعب توقع النتيجة

```
| let x = 10;  
| let y = 5;  
| let z = x + y; // Now z is 15
```

```
| let x = 10;  
| let y = "5";  
| let z = x + y; // Now z is "105"
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

سوء الفهم للآلة

يتم تخزين كافة الأرقام في جافاسكربت كأرقام الفاصلة العشرية ذات 64 بت (العائمات).

تواجه جميع لغات البرمجة، بما في ذلك جافاسكربت، صعوبات في تحديد قيم الفاصلة العشرية الدقيقة لأن معادلات الجبر في انشائها مختلفة:

```
| let x = 0.1;  
| let y = 0.2;  
| let z = x + y // the result in z will not be 0.3
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

لحل المشكلة أعلاه يساعد على الضرب والقسمة:

مثال

```
| let z = (x * 10 + y * 10) / 10; // z will be 0.3
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

كيف تقسيم الكود

ستسمح لك جافاسكربت بتقسيم العبارة إلى سطرين:

مثال 1

```
let x =  
"Abu Habib Al Husini *_*!";
```

let x =

"Abu Habib Al Husini *_*!" +

"Abu Habib Al Husini *_*!"

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

لكن تقسيم العبارة في منتصف النص لن ينجح

مثال 2

let x = "Abo Habib Al Hosini 😊😊"

Arabic!";

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

يجب عليك استخدام "خط مائل عكسي" إذا كان يجب عليك تقسيم عبارة : في نص

مثال 3

**let x = "Abo Habib Al Hosini 😊😊 **

Arabic!";

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

وضع فاصلة منقوطة في غير موضعها

بسبب وجود فاصلة منقوطة في غير مكانها، سيتم تنفيذ مقطع التعليمات X: البرمجية هذا بغض النظر عن قيمة

```
if (x == 19);  
{  
  // code block  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

اجراء عمليات بكلمة ريتورن من الاجراء

من سلوك جافاسكربت الافتراضي إغلاق العبارة تلقائيًا في نهاية السطر
وبسبب هذا، فإن هذين المثالين سيعودان بنفس النتيجة

مثال 1

```
function Husini(a) {  
  let power = 10  
  return a * power  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

مثال 2

```
function Husini(a) {  
  let power = 10;  
  return a * power;  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

ستسمح لك جافاسكربت أيضًا بتقسيم العبارة إلى سطرين
ولهذا السبب، سيعيد المثال 3 نفس النتيجة أيضًا

مثال 3

```
function Husini(a) {  
  let  
  power = 10;  
  return a * power;  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

ولكن ماذا سيحدث إذا قمت بتقسيم كلمة الإرجاع إلى سطرين مثل هذا

مثال 4

```
function Husini(a) {  
  let  
  power = 10;  
  return  
  a * power;  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

undefined! ستعود الوظيفة

لماذا؟ لأن جافاسكربت اعتقدت أنك تقصد انهيته سطر ريتورن وعمل اجراء
جديد:

مثال 5

```
function Husini(a) {  
  let  
  power = 10;  
  return;  
  a * power;  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

توضيح

إذا كانت العبارة غير كاملة مثل:

```
let
```

ستحاول جافاسكربت إكمال الكلمة من خلال قراءة السطر التالي:

```
power = 10;
```

ولكن بما أن هذا الكلمة كامل:

```
return
```

سوف يقوم جافاسكربت بإغلاقه تلقائيًا على النحو التالي:

```
return;
```

يحدث هذا لأن عبارات الإغلاق (الإنهاء) بفاصلة منقوطة أمر اختياري في جافاسكربت.

ستغلق جافاسكربت عبارة الإرجاع في نهاية السطر، لأنها عبارة كاملة.

لا تنقسم كلمة الإرجاع أبدًا.

جعل مفاتيح المصفوفة نصوص

تدعم العديد من لغات البرمجة المصفوفات ذات الفهارس النصية. تسمى المصفوفات ذات الفهارس النصية المصفوفات الترابطية (أو التجزئة). لا تدعم جافاسكربت المصفوفات ذات الفهارس النصية في جافاسكربت، تستخدم المصفوفات فهرس مرقمة فقط ولكن تستطيع استخدامها بطرق أخرى اقرأ على المرجع الكامل لجافاسكربت:

مثال

```
const hAbiB = [];  
hAbiB[0] = "Habib";  
hAbiB[1] = "Al Husini 🇸🇦🇸🇦🇸🇦";  
hAbiB[2] = 56;  
hAbiB.length; // hAbiB.length will return 3  
hAbiB[0]; // hAbiB[0] will return "Habib"
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

في جافاسكربت، تستخدم الكائنات الفهارس النصية.

إذا كنت تستخدم فهرسًا مسميًا، فعند الوصول إلى مصفوفة، ستعيد جافاسكربت تعريف المصفوفة إلى كائن قياسي

بعد إعادة التعريف التلقائي، ستنتج أساليب وخصائص المصفوفة نتائج غير محددة أو غير صحيحة:

مثال:

```
const hAbiB = [];  
hAbiB["firstName"] = "Habib";  
hAbiB["lastName"] = "Al Husini 🙄🙄🙄";  
hAbiB["age"] = 36;  
hAbiB.length; // hAbiB.length will return 0  
hAbiB[0]; // hAbiB[0] will return undefined
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

إنهاء البيانات بفاصلة

تعتبر الفواصل الزائدة في تعريف الكائن وال مصفوفة قانونية في ايكما 5

مثال: الكائن

```
hAbiB = {firstName:"Habib", lastName:"Al Husini 🙄🙄  
🙄", age:46,}
```

مثال: المصفوفة

```
hAbiB = [40, 100, 1, 5, 25, 10,];
```

!!! تحذير

سوف يتعطل متصفحات الفى 8.

بفواصل زائدة JSON لا يسمح

جسون:

```
hAbiB = {"firstName":"Habib", "lastName":"Al Husini", "age":46}
```

جسون:

```
hAbiB = [40, 100, 1, 5, 25, 10];
```

القيمة الغير محددة و الغير فارغة

يمكن أن تكون كائنات جافاسكربت ومتغيراتها وخصائصها **undefined** وأساليبيها

بالإضافة إلى ذلك، يمكن أن تحتوي كائنات جافاسكربت الفارغة على القيمة **null**.

وهذا يمكن أن يجعل اختبار ما إذا كان الكائن فارغًا أمرًا صعبًا بعض الشيء
يمكنك اختبار ما إذا كان الكائن موجودًا عن طريق اختبار ما إذا كان النوع هو **undefined**:

مثال:

| **if (typeof abib === "undefined")**

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

لأن هذا سيؤدي إلى **null** لكن لا يمكنك اختبار ما إذا كان الكائن موجودًا **undefined**: حدوث خطأ إذا كان الكائن

:غير صحيح

| **if (abib === null)**

null لحل هذه المشكلة، يجب عليك اختبار ما إذا كان الكائن ليس **undefined**. كذلك

:ولكن لا يزال من الممكن أن يؤدي هذا إلى حدوث خطأ

:غير صحيح

| **if (abib !== null && typeof abib !== "undefined")**

قبل أن تتمكن من اختبار **not undefined** ولهذا السبب، يجب عليك اختبار **not null**:

:صحيح

| **if (typeof abib !== "undefined" && abib !== null)**

أدوات

كيفية تسريع كود جافاسكربت

تقليل النشاط في الحلقات

غالبًا ما تستخدم الحلقات في البرمجة

لكل تكرار للحلقة **for** يتم تنفيذ كل عبلة في الحلقة، بما في ذلك عبلة العبارات أو المهام التي يمكن وضعها خارج الحلقة ستجعل الحلقة تعمل بشكل أسرع.

سيء:

```
for (let i = 0; i < arr.length; i++) {
```

كود أفضل:

```
let l = arr.length;  
for (let i = 0; i < l; i++) {
```

يصل الكود السيئ إلى خاصية طول المصفوفة في كل مرة يتم فيها تكرار الحلقة.

يصل الكود الأفضل إلى خاصية الطول خارج الحلقة ويجعل الحلقة تعمل بشكل أسرع.

تقليل الوصول إلى DOM

يعد الوصول إلى دوم بطيئًا جدًا، مقارنة بعبارات جافاسكربت الأخرى عدة مرات، فقم بالوصول إليه مرة DOM إذا كنت تتوقع الوصول إلى عنصر واحدة واستخدمه كمتغير محلي

مثال

```
const obj = document.getElementById("Habib");  
obj.innerHTML = "Abo Habib Al Hosini 🤪🤪";
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تقليل حجم DOM

اجعل عدد العناصر في دوم صغيرًا سيؤدي هذا دائمًا إلى تحسين تحميل الصفحة، وتسريع عرض (عرض الصفحة)، خاصة على الأجهزة الصغيرة.
مثل DOM ستستفيد كل محاولة للبحث في أصغر DOM من (getElementsByTagName)

تجنب المتغيرات غير الضرورية

لا تقم بإنشاء متغيرات جديدة إذا كنت لا تخطط لحفظ القيم في كثير من الأحيان يمكنك استبدال الكود مثل هذا

```
let fullName = firstName + " " + lastName;  
document.getElementById("Habib").innerHTML =  
fullName;
```

مع هذا

```
document.getElementById("Habib").innerHTML =  
firstName + " " + lastName;
```

تأخير تحميل

يتيح وضع الكود في أسفل نص الصفحة للمتصفح تحميل الصفحة أولاً أثناء تنزيل الكود، لن يبدأ المتصفح أي تنزيلات أخرى. بالإضافة إلى ذلك، قد يتم حظر جميع أنشطة التحليل والعرض.

أنه لا ينبغي للمتصفحات تنزيل أكثر من مكونين HTTP تحدد مواصفات بالتوازي.

في علامة الكود. تحدد سمة التأجيل أنه **defer="true"** البديل هو استخدامه يجب تنفيذ الكود بعد انتهاء تحليل الصفحة، ولكنها تعمل فقط مع الكود الخارجية.

إذا أمكن، يمكنك إضافة الكود إلى الصفحة عن طريق الكود، بعد تحميل الصفحة:

مثال

```
<script>  
window.onload = function() {
```

```
const element = document.createElement("script");
element.src = "Habib_Script.js";
document.body.appendChild(element);
};
</script>
```

تجنب الاستخدام كلمة with

الكلمة المحجوزة . له تأثير سلبي على السرعة. كما أنه **with** تجنب استخدام تشوش نطاقات جافاسكربت

المحجوزة غير مسموح بها في الوضع الصارم **with** الكلمة

جافاسكربت واهم الكلمات المحجوزة

في جافاسكربت، لا يمكنك استخدام هذه الكلمات المحجوزة كمتغيرات أو تسميات أو أسماء وظائف:

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	0	final
finally	float	for	function

goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	1
try	typeof	var	void
volatile	while	with	yield

الكلمات التي تحمل علامة * جديدة في ايكما 5 و6

احذر تمت إزالة هذه الكلمات المحجوزة

تمت إزالة الكلمات المحجوزة التالية من معيار ايكما 5/6:

abstract	boolean	byte	char
double	final	float	goto
int	long	native	short
synchronized	throws	transient	volatile

لا تستخدم هذه الكلمات كمتغيرات. لا يتمتع ايكما 5/6 بالدعم الكامل في جميع المتصفحات.

أشهر الكلمات المحجوزة المستخدمة

هذه الكلمات خاصة بجافاسكربت ولا تستخدمها في المتغيرات و يجب عليك أيضاً تجنب استخدامها في اسم الكائنات والخصائص والأساليب المضمنة في جافاسكربت:

Array	Date	eval	function
hasOwnProperty	Infinity	isFinite	isNaN
isPrototypeOf	length	Math	NaN
name	Number	Object	prototype
String	toString	undefined	valueOf

الكلمات المحجوزة

يجب عليك تجنب استخدام Java. غالباً ما يتم استخدام جافاسكربت مع كـمـرفـات جافاسكربت Java بعض كائنات وخصائص:

getClass java JavaArray javaClass
JavaObject JavaPackage

كلمات محجوزة أخرى

يمكن استخدام جافاسكربت كلغة برمجة في العديد من التطبيقات

HTML و Window: يجب عليك أيضاً تجنب استخدام اسم كائنات وخصائص

alert	all	anchor	anchors
area	assign	blur	button

checkbox	clearInterval	clearTimeout	clientInformation
close	closed	confirm	constructor
crypto	decodeURI	decodeURIComponent	defaultStatus
document	element	elements	embed
embeds	encodeURI	encodeURIComponent	escape
event	fileUpload	focus	form
forms	frame	innerHeight	innerWidth
layer	layers	link	location
mimeTypes	navigate	navigator	frames
frameRate	hidden	history	image
images	offscreenBuffering	open	opener
option	outerHeight	outerWidth	packages
pageXOffset	pageYOffset	parent	parseFloat
parseInt	password	pkcs11	plugin
prompt	propertyIsEnum	radio	reset
screenX	screenY	scroll	secure
select	self	setInterval	setTimeout
status	submit	taint	BiB

BiBarea	top	unescape	untaint
window			

معالجات أحداث HTML

بالإضافة إلى ذلك، يجب عليك تجنب استخدام اسم كافة معالجات أحداث HTML.

أمثلة:

onblur	onclick	onerror	onfocus
onkeydown	onkeypress	onkeyup	onmouseover
onload	onmouseup	onmousedown	onsubmit

الاستخدام الصارم

يحدد أنه يجب تنفيذ تعليمات جافاسكربت البرمجية في **"use strict"** الوضع الصارم.

باستخدام الوضع الصارم، على سبيل **المثال**، يمكنك عدم استخدام المتغيرات غير المعلنة.

يمكنك استخدام الوضع الصارم في جميع برامجك. يساعدك على كتابة تعليمات برمجية أنظف، مثل منعك من استخدام متغيرات غير معلنة.

هو مجرد تعبير نص . لن تقوم المتصفحات القديمة بإلقاء خطأ **"use strict"** إذا لم تفهمه.

الوصول إلى كل حرف في النصوص

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

مثال

```
var str = "Abu Habib Al Husini *_*";  
str.charAt(0); // returns H
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

:بالوصول إلى الخاصية عبر النصوص ES5 يسمح

مثال

```
var str = "Abu Habib Al Husini *_*";  
str[0]; // returns H
```

قد يكون الوصول إلى الخاصية على النص غير متوقع قليلاً

كتابة الكود على سطور متعددة

بنص حرفية عبر أسطر متعددة إذا تم الاستثناء باستخدام شرطة ES5 يسمح
مائلة عكسية:

مثال

```
"Abo Habib Al Hosini 😊😊 \\  
HABIB!";
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

قد لا تتمتع الطريقة \ بدعم عام

قد تتعامل المتصفحات القديمة مع المسافات الموجودة حول الشرطة المائلة
العكسية بشكل مختلف.

\. لا تسمح بعض المتصفحات القديمة بمسافات خلف الحرف

: الطريقة الأكثر أماناً لتفكيك نص حرفية هي استخدام إضافة النص

مثال

```
"Abo Habib Al Hosini 😊😊" +  
"HABIB!";
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

استخدام الكلمات المحجوزة كأسماء الخصائص

بالكلمات المحجوزة كأسماء خصائص ES5 يسمح

مثال الكائن

```
var obj = {name: "Habib", new: "yes"}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

ازالة المسافات من الجوانب

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

مثال

```
var str = "  Abu Habib Al Husini *_*!  ";  
alert(str.trim());
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة ()isArray

مما إذا كان الكائن عبارة عن مصفوفة (**isArray()**) تتحقق الطريقة

مثال

```
function Husini() {  
  var HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu  
Zedan"];  
  var x = document.getElementById("Habib");  
  x.innerHTML = Array.isArray(HaBiB);  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

جلب كل العناصر

دالة مرة واحدة لكل عنصر من عناصر (**forEach()**) تستدعي الطريقة المصفوفة.

مثال

```
var hAbiB = "";  
var Habib_Num = [45, 4, 9, 16, 25];  
Habib_Num.forEach(Husini);
```

```
function Husini(value) {  
  hAbiB = hAbiB + value + "<br>";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

تنفيذ اجراء على كل العناصر بدالة الخريطة

يقوم هذا **المثال** بضرب كل قيمة مصفوفة بمقدار 2

مثال

```
var Habib_Num1 = [45, 4, 9, 16, 25];  
var Habib_Num2 = Habib_Num1.map(Husini);
```

```
function Husini(value) {  
  return value * 2;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

الفلتر

ينشئ هذا **المثال** مصفوفة جديدة من عناصر ذات قيمة أكبر من 18

مثال

```
var Habib_Num = [45, 4, 9, 16, 25];  
var hosini_ = Habib_Num.filter(Husini);
```

```
function Husini(value) {  
  return value > 18;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة reduce

: يجد هذا المثال مجموع جميع الأرقام في مصفوفة

مثال

```
var Habib_Num1 = [45, 4, 9, 16, 25];  
var sum = Habib_Num1.reduce(Husini);
```

```
function Husini(total, value) {  
  return total + value;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة reduceRight

يُجد هذا **المثال** أيضًا مجموع جميع الأرقام في المصفوفة

مثال

```
var Habib_Num1 = [45, 4, 9, 16, 25];  
var sum = Habib_Num1.reduceRight(Husini);
```

```
function Husini(total, value) {  
  return total + value;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

داله اخرى للعمليات على المصفوفة

يتحقق هذا **المثال** مما إذا كانت جميع القيم أكبر من 18

مثال

```
var Habib_Num = [45, 4, 9, 16, 25];  
var All_Habib_Num_hosini_ = Habib_Num.every(Husini);
```

```
function Husini(value) {  
  return value > 18;  
}
```


دالة some

يتحقق هذا **المثال** مما إذا كانت بعض القيم أكبر من 18

مثال

```
var Habib_Num = [45, 4, 9, 16, 25];  
var All_Habib_Num_hosini_ = Habib_Num.some(Husini);  
  
function Husini(value) {  
  return value > 18;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة IndexOf

ابحث في مصفوفة عن قيمة عنصر وقم بإرجاع موضعها.

مثال

```
var HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
var a = HaBiB.indexOf("Abu Habib");
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة lastIndexOf ()

ولكن يبحث من نهاية المصفوفة، **indexOf()** هو نفسه **lastIndexOf()**.

مثال

```
var HaBiB= ["Al-Husini", "Osman", "Abu Habib", "Abu Zedan"];  
var a = HaBiB.lastIndexOf("Abu Habib");
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

JSON.parse()

هو تلقي الكلمات من خادم الويب JSON الاستخدام الشائع لـ
تحويل أنك تلقيت هذه النص النصية من خادم الويب

```
'{"name":"Habib", "age":30, "city":"Hosini}"
```

لتحويل النص إلى كائن `JSON.parse()` يتم استخدام وظيفة جافاسكربت جافاسكربت:

```
var obj = JSON.parse('{"name":"Habib", "age":30, "city":"Hosini}');
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

JSON.stringify()

هو إرسال الكلمات إلى خادم الويب JSON الاستخدام الشائع لـ عند إرسال الكلمات إلى خادم الويب، يجب أن تكون الكلمات عبارة عن نص .

تخيل أن لدينا هذا الكائن في جافاسكربت

```
var obj = {name:"Habib", age:30, city:"Hosini"};
```

لتحويلها إلى نص `JSON.stringify()` استخدم وظيفة جافاسكربت

```
var myJSON = JSON.stringify(obj);
```

JSON ستكون النتيجة نص تتبع تدوين

:الآن نص نصية، وجاهزاً لإرساله إلى الخادم myJSON أصبح

مثال

```
var obj = {name:"Habib", age:30, city:"Hosini"};
var myJSON = JSON.stringify(obj);
document.getElementById("Habib").innerHTML =
myJSON;
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

كيف جلب التاريخ.الآن

تُرجع عدد المللي ثانية منذ تاريخ الصفر (1 يناير 1970، `Date.now()`، 00:00:00 بالتوقيت العام).

مثال

```
var timInMSs = Date.now();
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

`Date` التي يتم إجراؤها على `getTime()` تُرجع نفس طريقة `Date.now()` كائن.

() ISOString تحويل التاريخ إلى

بتحويل كائن تحويل التاريخ إلى نص ، **toISOString()** تقوم الطريقة
ISO: باستخدام التنسيق القياسي

مثال

```
const d = new Date();  
document.getElementById("Habib").innerHTML =  
d.toISOString();
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

() JSON تحويل التاريخ إلى

JSON. يحول كائن تحويل التاريخ إلى نص ، منسقة كتاريخ **toJSON()**
ISO-8601: YYYY-MM-DDTHH:mm:ssZ لها نفس تنسيق معيار JSON تواريخ
DDTHH:mm:ssZ:

مثال

```
d = new Date();  
document.getElementById("Habib").innerHTML =  
d.toJSON();
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

شاهد هذه الأمثلة

تحديد أساليب الكائن باستخدام بناء جملة يشبه الحصول على ES5 يتيح لك خاصية أو تعيينها.

fullName: لخاصية تسمى getter ينشئ هذا المثال أداة

مثال

```
// Create an object:
```

```
var hAbiB = {  
  firstName: "Habib",  
  lastName : "Al Husini 😊😊😊",  
  get fullName() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

```
// Display data from the object using a getter:
```

```
document.getElementById("Habib").innerHTML = hAbi  
B.fullName;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

:ينشئ هذا المثال أداة ضبط وأداة إحضار لخاصية اللغة

مثال

```
var hAbiB = {  
  firstName: "Habib",  
  lastName : "Al Husini 😊😊😊",
```

```
language : "NO",
get lang() {
  return this.language;
},
set lang(value) {
  this.language = value;
}
};
```

// Set an object property using a setter:

```
hAbiB.lang = "en";
```

// Display data from the object using a getter:

```
document.getElementById("Habib").innerHTML = hAbiB.lang;
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

يستخدم هذا المثال أداة ضبط لتأمين تحديثات الأحرف الكبيرة للغة:

مثال

```
var hAbiB = {
  firstName: "Habib",
  lastName : "Al Husini 😊😊😊",
  language : "NO",
  set lang(value) {
    this.language = value.toUpperCase();
  }
};
```

```
// Set an object property using a setter:
```

```
hAbiB.lang = "en";
```

```
// Display data from the object:
```

```
document.getElementById("Habib").innerHTML = hAbiB.language;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

Object.defineProperty()

Object.defineProperty() هي طريقة كائن جديدة في ES5.

يتيح لك تحديد خاصية كائن و/أو تغيير قيمة الخاصية و/أو كلمات التعريف.

مثال

```
// Create an Object:
```

```
var hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 🇸🇦🇸🇦🇸🇦",  
  language: "Arabic",  
};
```

```
// Change a Property:
```

```
Object.defineProperty(hAbiB, "language", {  
  value: "EN",
```



```
writable : true,  
enumerable : true,  
configurable : true  
});
```

```
// Enumerate Properties
```

```
var hAbiB = "";  
for (var x in hAbiB) {  
  hAbiB += hAbiB[x] + "<br>";  
}  
document.getElementById("Habib").innerHTML = hAbiB;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

:المثال التالي هو نفس الكود، إلا أنه يخفي خاصية اللغة من التعداد

مثال

```
// Create an Object:
```

```
var hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  language: "NO",  
};
```

```
// Change a Property:
```

```
Object.defineProperty(hAbiB, "language", {  
  value: "EN",  
  writable : true,
```

```
enumerable : false,  
configurable : true  
});
```

```
// Enumerate Properties
```

```
var hAbiB = "";  
for (var x in hAbiB) {  
  hAbiB += hAbiB[x] + "<br>";  
}  
document.getElementById("Habib").innerHTML = hAbiB;
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

لتأمين تحديثات الأحرف الكبيرة **getter** ينشئ هذا **المثال** أداة ضبط وأداة للغة:

مثال

```
/// Create an Object:  
var hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  language: "NO"  
};
```

```
// Change a Property:
```

```
Object.defineProperty(hAbiB, "language", {  
  get : function() { return language },  
  set : function(value) { language = value.toUpperCase()}}
```

```
});
```

```
// Change Language
```

```
hAbiB.language = "en";
```

```
// Display Language
```

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

```
hAbiB.language;
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

ربط الادالة

الطريقة، يمكن للكائن استعارة طريقة من كائن آخر (`bind()`) باستخدام

يقوم هذا **المثال** بإنشاء كائنين (شخص وعضو)

: يستعير كائن العضو طريقة الاسم الكامل من كائن العنصر

مثال

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

```
const member = {
```

```
firstName:"Habib",
lastName: "Nilsen",
}
```

```
let fullName = hAbiB.fullName.bind(member);
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

فواصل زائدة

بفواصل زائدة في بيانات الكائنات والمصفوفات ES5 يسمح

مثال الكائن

```
hAbiB = {
  firstName: "Habib",
  lastName: " Al Husini 😊😊😊",
  age: 46,
}
```

مثال المصفوفة

```
hAbiB = [
  1,
  5,
  10,
  25,
```

```
40,  
100,  
];
```

!!! تحذير

لا يسمح جسون بفواصل زائدة

كائنات جسون:

// Allowed:

```
var hAbiB = '{"firstName":"Habib", "lastName":"Al  
Husini 😊😊😊", "age":46}'  
JSON.parse(hAbiB)
```

// Not allowed:

```
var hAbiB = '{"firstName":"Habib", "lastName":"Al  
Husini 😊😊😊", "age":46,}'  
JSON.parse(hAbiB)
```

JSON: مصفوقات

// Allowed:

```
hAbiB = [40, 100, 1, 5, 25, 10]
```

// Not allowed:

```
hAbiB = [40, 100, 1, 5, 25, 10,]
```

إعادة الإعلان

المحجوزة الإعلان عن متغير بنطاق الكتلة **let** تتيح لك الكلمة

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الثابت في

المحجوزة الإعلان عن ثابت (متغير جافاسكربت بقيمة **const** تتيح لك الكلمة ثابتة).

باستثناء أن القيمة لا يمكن تغييرها، **let** الثوابت تشبه متغيرات

مثال

```
var x = 10;  
// Here x is 10
```

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

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

وظائف مختصرة

تسمح وظائف مختصرة بتركيب جملة قصيرة لكتابة تعبيرات الوظائف المحجوزة، **return** الكلمة المحجوزة، والكلمة **function** لا تحتاج إلى والأقواس المتعرجة.

مثال

```
// ES5  
var x = function(x, y) {  
  return x * y;  
}
```

```
// ES6  
const x = (x, y) => x * y;
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

وهي ليست مناسبة تمامًا. **this** وظائف مختصرة ليس لها وظائفها الخاصة. لتحديد أساليب الكائنات.

لا يتم رفع وظائف مختصرة. ويجب تعريفها قبل استخدامها.

لأن تعبير الدالة يكون **var** أكثر أماناً من الاستخدام **const** يعد الاستخدام دائماً قيمة ثابتة

الكلمة المحجوزة والأقواس المتعرجة فقط إذا كانت الدالة **return** يمكنك حذف عبارة واحدة. ولهذا السبب، قد يكون من العادات الجيدة الاحتفاظ بها دائماً

مثال

```
const x = (x, y) => { return x * y };
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

عامل الانتشار

يقوم عامل التشغيل ... بتوسيع العنصر القابل للتكرار (مثل المصفوفة) إلى المزيد من العناصر

مثال

```
const HaBiB1 = ["Habib", "Hamza", "Abu Habib Al-Husini"];  
const HaBiB2 = ["Habib", "Hamza", [..."Abu Habib Al-Husini"]];
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

يمكن استخدام عامل التشغيل ... لتوسيع كائن قابل للتكرار إلى المزيد من البرامتر لاستدعاءات الوظائف

مثال

```
const Habib_Num = [23,55,21,87,56];  
let maxValue = Math.max(...Habib_Num);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

For/Of حلقة

عبر قيم الكائنات القابلة للتكرار **for/of** تتكرر عبارة جافاسكربت

يتيح لك التكرار عبر هياكل الكلمات القابلة للتكرار مثل المصفوفات **for/of** والنصوص والخرائط وقوائم العقد والمزيد
:على بناء الجملة التالي **for/of** تحتوي الحلقة

```
for (variable of iterable) {  
    // code block to be executed  
}
```

متغير - لكل تكرار يتم تعيين قيمة الخاصية التالية للمتغير. يمكن الإعلان عن **var** أو **let** بـ **const** المتغير

قابل للتكرار - كائن له خصائص قابلة للتكرار

التكرار فوق مصفوفة

مثال

```
const HaBiB= ["Abu Habib Al-Husini", "Hamza", "Mini"];  
let BiB = "";  
  
for (let x of HaBiB) {  
  BiB += x + " ";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

التكرار على نص

مثال

```
let language = "javascript";  
let BiB = "";  
  
for (let x of language) {  
  BiB += x + " ";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

الإعلان عن كائن الخريطة

تعد القدرة على استخدام كائن كمفتاح إحدى ميزات الخريطة المهمة.

مثال

```
const HaBiB= new Map([\n  ["Abu Habibs", 500],\n  ["Al-Husinis", 300],\n  ["Osmans", 200]\n]);
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

تعرف على المزيد حول كائنات الخريطة، والفرق بين الخريطة والمصفوفة، في الفصل .

مجموعات

مثال

```
// Create a Set\nconst Hbib = new Set();\n\n// Add some values to the Set\nHbib.add("a");\nHbib.add("b");\nHbib.add("c");
```

فئات

فئات جافاسكربت هي قوالب لكائنات جافاسكربت.

لإنشاء كلاس **class** استخدم الكلمة المحجوزة

constructor(): أضف دائمًا طريقة باسم

بناء الجملة

```
class ClassName {  
  constructor() { ... }  
}
```

مثال

```
class $_Habib{  
  constructor(name, year) {  
    this.name = name;  
    this.year = year;  
  }  
}
```

"\$_Habib" يقوم **المثال** أعلاه بإنشاء كلاس تسمى

"يحتوي الفصل على خاصيتين أوليتين: "الاسم" و"السنة"

باستخدام الكلاسات

:عندما يكون لديك كلاس، يمكنك استخدامه لإنشاء كائنات

مثال

```
const my$_Habib1 = new $_Habib("Mohamid*", 2014);  
const my$_Habib2 = new $_Habib("Omar ibn Al-  
khattab", 2019);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

كائن الوعود في

هو كائن جافاسكربت يربط بين "إنتاج التعليمات البرمجية" Promise() ال
و"استهلاك التعليمات البرمجية".

يمكن أن يستغرق "إنتاج التعليمات البرمجية" بعض الوقت ويجب أن ينتظر
"استهلاك التعليمات البرمجية" النتيجة.

انشاء الجملة

```
const Habib2romise  
= new Promise(function(myResolve, myReject) {  
// "Producing Code" (May take some time)
```

```
myResolve(); // when successful  
myReject(); // when error  
});
```

```
// "Consuming Code" (Must wait for a fulfilled Promise).  
Habib2romise.then(  
function(value) { /* code if successful */ },  
function(error) { /* code if some error */ }  
);
```

Promise() مثال باستخدام ال

```
const Habib2romise  
= new Promise(function(myResolve, myReject) {  
setTimeout(function() { myResolve("I love  
You !!"); }, 3000);  
});
```

```
Habib2romise.then(function(value) {  
document.getElementById("Habib").innerHTML =  
value;  
});
```

نوع الإجراء أو الكود

أو String أو Number كود جافاسكربت هو نوع كلمات بدائي تمامًا مثل Boolean.

إنه يمثل معرفًا "مخفيًا" فريدًا لا يمكن لأي كود آخر الوصول إليه عن طريق الخطأ.

على سبيل المثال، إذا أراد المبرمج حسينى المختلفون إضافة خاصية إلى كائن شخص ينتمي إلى كود جهة خارجية، فيمكنهم خلط قيم hAbiB.id بعضهم البعض.

:يؤدي استخدام الاجراء () لإنشاء معرفات فريدة إلى حل هذه المشكلة

مثال

```
const hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  age: 500,  
  eyeColor: "red😊"  
};
```

```
let id = Symbol('id');  
hAbiB[id] = 140353;
```

```
// Now hAbiB[id] = 140353
// but hAbiB.id is still undefined
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

ملحوظة

الرموز دائما فريدة من نوعها.

إذا قمت بإنشاء كودين بنفس الوصف فسيكون لهما قيم مختلفة

```
Symbol("id") == Symbol("id"); // false
```

قيم البرمترات الافتراضية

لمعاملات الوظيفة بالحصول على قيم افتراضية ES6 يسمح

مثال

```
function Husini(x, y = 10) {
  // y is 10 if not passed or undefined
  return x + y;
}
Husini(5); // will return 15
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

البرامتر غير محدود الأعداد

تسمح البرامتر الباقية (...) للدالة بمعاملة عدد غير محدد من البرامتر كمصفوفة:

مثال

```
function sum(...args) {  
  let sum = 0;  
  for (let arg of args) sum += arg;  
  return sum;  
}
```

```
let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة تتضمن

كانت النص تحتوي على قيمة محددة، **true** تُرجع الطريقة إذا **includes()** وإلا **false**:

مثال

```
let BiB = "Abu Habib Al Husini *_*, welcome to the  
universe."  
BiB.includes("Arabic") // Returns true
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة startsWith

كانت النص النصية تبدأ بقيمة محددة، **true** تُرجع الطريقة إذا **startsWith()** وإلا **false**:

مثال

```
let BiB = "Abu Habib Al Husini *_*, welcome to the universe.";
```

```
BiB.startsWith("Abo Habib Al Hosini 😊😊") // Returns true
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

String.endsWith()

false: انتهت نص بقيمة محددة، وإلا **true** تُرجع الطريقة إذا **endsWith()**:

مثال

```
var BiB = "Abu Bakr Al-Siddiq";
```

```
BiB.endsWith("Al Husini 😊😊😊") // Returns true
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

.Array.from()

إرجاع كائن مصفوفة من أي كائن له خاصية **Array.from()** تقوم الطريقة الطول أو أي كائن قابل للتكرار.

مثال

: إنشاء مصفوفة من نص

```
Array.from("ABCDEFGH") // Returns [A,B,C,D,E,F,G,H]
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

جلب المفاتيح

مع مفاتيح المصفوفة **Array Iterator** إرجاع كائن **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();
```

```
let BiB = "";
```

```
for (let x of keys) {
```

```
BiB += x + "<br>";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

طرق البحث

قيمة عنصر ال مصفوفة الأول الذي يمرر دالة اختبار **find()** تُرجع الطريقة
:يبحث هذا **المثال** 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 الثالث ابو حبيب الحسيني

الدالة () findIndex

بإرجاع فهرس عنصر ال مصفوفة الأول الذي يمرر **findIndex()** تقوم الطريقة
بوظيفة الاختبار

:يعثر هذا **المثال** 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 جافاسكريبت الجزء الثالث ابو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة العنصر
- فهرس البند
- المصفوفة نفسها

طرق الرياضيات

Math: الطرق التالية إلى كائن ES6 أضاف

- **Math.trunc()**
- **Math.sign()**
- **Math.cbrt()**
- **Math.log2()**
- **Math.log10()**

طريقة Math.trunc()

Math.trunc(x): إرجاع الجزء الصحيح من x :

مثال

```
Math.trunc(4.9); // returns 4
Math.trunc(4.7); // returns 4
Math.trunc(4.4); // returns 4
Math.trunc(4.2); // returns 4
Math.trunc(-4.2); // returns -4
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

طريقة Math.sign()

Math.sign(x): سالبة أو فارغة أو موجبة x تُرجع إذا كانت x سالبة أو فارغة أو موجبة:

مثال

```
Math.sign(-4); // returns -1
Math.sign(0); // returns 0
Math.sign(4); // returns 1
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

طريقة Math.cbrt()

Math.cbrt(x): إرجاع الجذر التكعيبي لـ x :

مثال

```
Math.cbrt(8); // returns 2  
Math.cbrt(64); // returns 4  
Math.cbrt(125); // returns 5
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

طريقة Math.log2()

Math.log2(x): إرجاع اللوغاريتم الأساسي 2 لـ x :

مثال

```
Math.log2(2); // returns 1
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

طريقة Math.log10()

Math.log10(x): إرجاع اللوغاريتم 10 لـ x :

مثال

```
Math.log10(10); // returns 1
```

خصائص كلاس الأرقام

:الخصائص التالية إلى كائن الرقم ES6 أضاف

- **EPSILON**
- **MIN_SAFE_INTEGER**
- **MAX_SAFE_INTEGER**

مثال إبسيلون

```
let x = Number.EPSILON;
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

MIN_SAFE_INTEGER مثال

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

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

MAX_SAFE_INTEGER مثال

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

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

طرق الأرقام

طريقتين جديدتين إلى كائن الرقم ES6 أضاف:

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

طريقة `Number.isInteger()`

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

مثال

```
Number.isInteger(10); // returns true  
Number.isInteger(10.5); // returns false
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

طريقة `Number.isSafeInteger()`

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

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

مثال

```
Number.isSafeInteger(10); // returns true  
Number.isSafeInteger(12345678901234567890); //  
returns false
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الأعداد الصحيحة الآمنة هي جميع الأعداد الصحيحة من -(53 - 2) إلى +(2 - 53).

هذا آمن: 9007199254740991. هذا غير آمن: 9007199254740992

أساليب عامة جديدة

طريقتين جديدتين للأرقام العامة ES6 أضاف:

- **isFinite()**
- **isNaN()**

isFinite() طريقة

NaN أو **Infinity** إذا كانت الوسيطة **false** تُرجع الطريقة العامة **isFinite()**.

true: وإلا فإنه يعود

مثال

```
isFinite(10/0); // returns false  
isFinite(10/1); // returns true
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

isNaN() طريقة

وإلا فإنه **NaN**. إذا كانت الوسيطة **true** تُرجع الطريقة العامة **isNaN()** **false**: يعود

مثال

```
isNaN("Abo Habib Al Hosini 😊😊"); // returns true
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

إلِدخالات والتكرارات

مثال

:أنشئ حلقة المصفوفة، ثم كرهه 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 الثالث ابو حبيب الحسيني

مع أزواج Array Iterator بإرجاع كائن **entries()** تقوم الطريقة
:المفتاح/القيمة

```
[0, "موز"]  
[1, "برتقال"]  
[2, "تفاحة"]  
[3, "مانجو"]
```

. لا تغير الطريقة المصفوفة الأصلية **entries()**.

وحدات

:يتم استيراد الوحدات بطريقتين مختلفتين

الاستيراد من التصديرات النصية

hAbiB.js: استيراد التصديرات النصية من الملف

```
import { name, age } from "./hAbiB.js";
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

الاستيراد من التصديرات الافتراضية

message.js: استيراد تصدير افتراضي من الملف

import message from "./message.js";

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

ايكما 2016 ميزات جديدة في

يقدم هذا الفصل الميزات الجديدة في ايكما 2016

- جافاسكربت الأسي (**)
- مهمة جافاسكربت الأسي (**=)
- التضمين بطرق سهلة في مصفوفة جافاسكربت

عامل الأس

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

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث 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 الثالث ابو حبيب الحسيني

مهمة الأس

يقوم عامل الإسناد الأسّي (****=**) برفع قيمة المتغير إلى قوة المعامل الأيمن

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

يتم دعم عامل الأس في جميع المتصفحات الحديثة منذ مارس 2017

تابع التضمن للمصفوفة

إلى المصفوفات **Array.includes** تم تقديم ايكما 2016
يتيح لنا ذلك التحقق من وجود عنصر في المصفوفة

مثال

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

```
HaBiB.includes("Abu Zedan");
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

مدعوم في جميع المتصفحات الحديثة منذ أغسطس **Array.includes**
2016:

ES6 (2015) و ES5 (2009) أرقام إصدارات جافاسكربت الأصلية كانت

نص الربط

الربط في **padEnd** ولدعم **padStart**: أضاف ايكما 2017 طريقتين للنص
بداية النص وفي نهايتها

أمثلة

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

يتم دعم ربطة نص جافاسكربت في جميع المتصفحات الحديثة منذ أبريل 2017:

إدخالات الكائن

الطريقة إلى الكائنات **Object.entries()** أضاف ايكما 2017.

إرجاع مصفوفة من أزواج المفتاح/القيمة في كائن **Object.entries()**

مثال

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


let BiB = Object.entries(hAbiB);

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

Object.entries() يجعل من السهل استخدام الكائنات في الحلقات

مثال

```
const HaBiB= {Al-Husinis:300, Osmans:200, Abu  
Habibs:500};
```

```
let BiB = "";  
for (let [fruit, value] of Object.entries(HaBiB)) {  
  BiB += fruit + ": " + value + "<br>";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

كما أنه يجعل من السهل تحويل الكائنات إلى خرائط

مثال

```
const HaBiB= {Al-Husinis:300, Osmans:200, Abu  
Habibs:500};
```

```
const myMap = new Map(Object.entries(HaBiB));
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

مدعوم في جميع المتصفحات الحديثة منذ مارس 2017:

قيم الكائنات

ولكنها تُرجع مصفوفة ذات `Object.entries()`، تشبه `Object.values()` بعد واحد لقيم الكائنات:

مثال

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

```
let BiB = Object.values(hAbiB);
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

مدعوم في جميع المتصفحات الحديثة منذ مارس 2017 `Object.values()`

وظائف غير المتزامنة

في انتظار المهلة للتنفيذ

```
async function myDisplay() {  
  let Habib2romise = new Promise(function(myResolve,  
  myReject) {
```

```
setTimeout(function() { myResolve("I love  
You !!"); }, 3000);  
});
```

```
document.getElementById("Habib").innerHTML = await  
Habib2romise;  
}
```

```
myDisplay();
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

التكرار غير المتزامن

أضاف ايكما 2018 حلقات غير متزامنة وقابلة للتكرار

الكلمة المحجوزة **await** مع العناصر التكرارية غير المتزامنة، يمكننا استخدام الحلقات **for/of** في

مثال

```
for await () {}
```

يتم دعم التكرار غير المتزامن لـ جافاسكربت في جميع المتصفحات الحديثة
منذ يناير 2020

وأخيراً كائن ال Promise()

من Promise ينهي ايكما 2018 التنفيذ الكامل لكائن خلال **Promise.finally**:

مثال

```
let Habib2romise = new Promise();
```

```
Habib2romise.then();  
Habib2romise.catch();  
Habib2romise.finally();
```

مدعوم في جميع المتصفحات الحديثة منذ نوفمبر 2018 **Promise.finally**

جلب الخصائص بالطريقة الحديثة

. أضاف ايكما 2018 جلب بأسلوب سهل في
يتيح لنا ذلك إزالة كائن وجمع البقايا في كائن جديد

مثال

```
let { x, y, ...z } = { x: 1, y: 2, a: 3, b: 4 };  
x; // 1  
y; // 2  
z; // { a: 3, b: 4 }
```

يتم دعم خصائص بقاء الكائنات في جميع المتصفحات الحديثة منذ يناير 2020:

مواضيع

لإنشاء نصوص **Web Workers API** في جافاسكربت، يمكنك استخدام الرسائل.

تُستخدم مؤشرات الترابط العاملة لتنفيذ التعليمات البرمجية في الخلفية حتى يتمكن البرنامج الرئيسي من مواصلة التنفيذ.

تعمل مؤشرات الترابط العاملة في وقت واحد مع البرنامج الرئيسي. التنفيذ المتزامن لأجزاء مختلفة من البرنامج يمكن أن يوفر الوقت.

الذاكرة المشتركة

الذاكرة المشتركة هي ميزة تسمح للنصوص (أجزاء مختلفة من البرنامج) بالوصول إلى نفس الكلمات وتحديثها في نفس الذاكرة.

بدلاً من تمرير الكلمات بين نصوص العمليات، يمكنك تمرير كائن **SharedArrayBuffer** الذي يشير إلى الذاكرة حيث يتم حفظ الكلمات.

SharedArrayBuffer

مخزناً مؤقتاً للكلمات الثنائية الأولية ذات **SharedArrayBuffer** يمثل كائن **ArrayBuffer** طول ثابت مشابهاً لكائن

دالة () TrimStart

إلى جافاسكربت **String trimStart()** طريقة **ES2019** أضاف.

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

مثال

```
let Hosini1 = " Abu Habib Al Husini *_*! ";  
let Hosini2 = Hosini1.trimStart();
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

دالة TrimEnd ()

إلى جافاسكربت **trimEnd()** String طريقة ES2019 أضاف

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

مثال

```
let Hosini1 = " Abu Habib Al Husini *_*! ";  
let Hosini2 = Hosini1.trimEnd();
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

fromEntries() كائن

إلى جافاسكربت **fromEntries()** طريقة الكائن ES2019 أضاف بإنشاء كائن من أزواج المفاتيح/القيمة القابلة **fromEntries()** تقوم الطريقة للتكرار.

مثال

```
const HaBiB= [  
  ["Abu Habibs", 300],  
  ["Hamza", 900],  
  ["Al-Husinis", 500]  
];
```

```
const abib = Object.fromEntries(HaBiB);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

كشف الأخطاء

يمكنك حذف برامتر الالتقاط إذا لم تكن بحاجة إليها ES2019 من

مثال

قبل عام 2019

```
try {  
  // code  
} catch (err) {  
  // code  
}
```

بعد عام 2019

```
try {  
  // code  
} catch {  
  // code  
}
```

يتم دعم ربط الالتقاط الاختياري في جميع المتصفحات منذ يناير 2020

دالة flat()

إلى جافاسكريبت **flat()** Array طريقة ES2019 أضاف

. بإنشاء مصفوفة جديدة عن طريق تسوية مصفوفة متداخلة **flat()** تقوم الطريقة

مثال

```
const Habib_myArr = [[1,2],[3,4],[5,6]];  
const newArr = Habib_myArr.flat();
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

الدالة () flatMap

إلى جافاسكربت (**flatMap()** Array طريقة ES2019 أضاف

أولاً بتعيين جميع عناصر المصفوفة ثم تقوم بإنشاء (**flatMap()** تقوم الطريقة مصفوفة جديدة عن طريق تسوية المصفوفة

مثال

```
const Habib_myArr = [1, 2, 3, 4, 5, 6];  
const newArr = Habib_myArr.flatMap(x => x * 2);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الفرز

(**sort()** بمراجعة طريقة المصفوفة ES2019 قام

قبل عام 2019، كانت المواصفات تسمح بخوارزميات فرز غير مستقرة مثل QuickSort.

:يجب أن تستخدم المتصفحات خوارزمية فرز مستقرة، ES2019 بعد

عند فرز العناصر على قيمة ما، يجب أن تحافظ العناصر على موقعها النسبي مع العناصر الأخرى التي لها نفس القيمة

مثال

```
const Habib_myArr = [  
  {name:"X00",price:100 },  
  {name:"X01",price:100 },  
  {name:"X02",price:100 },  
  {name:"X03",price:100 },  
  {name:"X04",price:110 },  
  {name:"X05",price:110 },  
  {name:"X06",price:110 },  
  {name:"X07",price:110 }  
];
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

في المثال أعلاه، عند الفرز على السعر، لا يُسمح للنتيجة بالخروج بالأسماء
:في موضع نسبي آخر مثل هذا

```
X01 100  
X03 100  
X00 100  
X03 100  
X05 110  
X04 110  
X06 110  
X07 110
```

الوظيفة JSON.stringify

JSON.stringify() بمراجعة طريقة ES2019 قام

من تقييد الأحرف المشفرة بـ JSON قبل عام 2019، لم يتمكن

مثال

```
let BiB = JSON.stringify("\u26D4");
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

على نقاط ترميز JSON.stringify() أدى استخدام ES2019 قبل معطلة Unicode إلى إرجاع أحرف (U+DFFF إلى U+D800 من) UTF-8 مثل ❏❏❏.

UTF-8 بعد هذه المراجعة، يتم تحويل النصوص ذات نقاط كود JSON.parse() باستخدام JSON.stringify() بأمان

الرموز الفاصلة

في (\u2028 و \u2029) يُسمح الآن بفواصل الأسطر ورموز فواصل الفقرات نص حرفية.

قبل عام 2019، كان يتم التعامل معها على أنها نقاط إنهاء للخطوط مما أدى إلى حدوث استثناءات للأخطاء

مثال

```
// This is valid in ES2019:  
let BiB = "\u2028";
```

ملحوظة

قواعد متساوية JSON الآن، لدى جافاسكربت و

قبل ES2019:

BiB = JSON.parse("\u2028") سيتم تحليله إلى "

BiB = "\u2028" قد يعطي خطأ في بناء الجملة

الكلاس () String

toString() بمراجعة طريقة الوظيفة ES2019 قام

بإرجاع نص تمثل الكود المصدري للوظيفة toString() تقوم الطريقة

بإرجاع الكود المصدري toString() اعتباراً من عام 2019، يجب أن تقوم للوظيفة بما في ذلك التعليقات والمسافات وتفاصيل بناء الجملة

قبل عام 2019، كانت المتصفحات المختلفة تعرض أشكالاً مختلفة من الوظيفة (مثل بدون تعليقات ومسافات). اعتباراً من عام 2019، يجب إرجاع الوظيفة تماماً كما هي مكتوبة

مثال

```
function Husini(p1, p2) {  
  return p1 * p2;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

دالة () matchAll

لم تكن هناك طريقة نص يمكن استخدامها للبحث عن جميع ES2020 قبل . تكرارات النص في النص

مثال

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

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

وإلا فسيتم ، (g) إذا كانت البوامتر تعبيراً عادياً، فيجب تعيين العلامة العامة
TypeError طرح خطأ

مثال

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

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

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

مثال

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

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

عامل الدمج الفارغ (??)

null يقوم عامل **??** التشغيل بإرجاع الوسيطة الأولى إذا لم تكن فارغة أو **undefined**.

وإلا فإنه يعود الثاني

مثال

```
let name = null;  
let BiB = "missing";  
let result = name ?? BiB;
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

مشغل التسلسل الاختياري (?.)

undefined إذا كان الكائن **undefined** يُرجع عامل التسلسل الاختياري **undefined** (بدلاً من إلقاء خطأ) أو **null**.

مثال

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

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

:المشغل **?.** = مدعوم في جميع المتصفحات منذ مارس 2020

= && المشغل

يتم استخدام عامل التشغيل المنطقي والتعيين بين قيمتين.
يتم تعيين القيمة الثانية، **true** إذا كانت القيمة الأولى هي

مثال منطقي والواجب

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

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

:المشغل **&&=** = مدعوم في جميع المتصفحات منذ سبتمبر 2020

= || المشغل

يتم استخدام عامل التشغيل المنطقي أو التعيين بين قيمتين.
يتم تعيين القيمة الثانية، **false** إذا كانت القيمة الأولى هي

مثال منطقي أو التنازل

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

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

:المشغل ||= مدعوم في جميع المتصفحات منذ سبتمبر 2020

ال ?? = المشغل

بين قيمتين Nullish Coalescing يتم استخدام عامل التعيين. فسيتم تعيين القيمة الثانية، **null** أو **undefined** إذا كانت القيمة الأولى هي

مثال على مهمة الدمج الفارغة

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

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

:المشغل ??= مدعوم في جميع المتصفحات منذ سبتمبر 2020

البحث والاستبدال لكل التطابقات دفعة واحدة

`replaceAll()`: طريقة النص ES2021 قدم

مثال

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

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تحديد تعبير عادي بدلاً من نص ليتم `replaceAll()` تتيح لك الطريقة استبدالها.

وإلا فسيتم `(g)` إذا كانت البوامة تعبيراً عادياً، فيجب تعيين العلامة العامة `TypeError` طرح خطأ.

مثال

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

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

استخدام الفاصلة (_)

:الفاصل الرقمي (_) لجعل الأرقام أكثر قابلية للقراءة ES2021 أدخل

مثال

```
const num = 1_000_000_000;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الفاصل الرقمي مخصص للاستخدام المرئي فقط.

مثال

```
const num1 = 1_000_000_000;  
const num2 = 1000000000;  
(num1 === num2);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

يمكن وضع الفاصل الرقمي في أي مكان في الرقم:

مثال

```
const num1 = 1_2_3_4_5;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

ملحوظة

لا يُسمح بالفاصل الرقمي في بداية الرقم أو نهايته.

في جافاسكربت، يمكن للمتغيرات فقط أن تبدأ بـ

الفاصل الرقمي مدعوم في جميع المتصفحات منذ يناير 2020

النهاية المأسوية للإنترنت إكسبلورر

لا ينصح باستخدام هذا المتصفح القديم بسبب رفع الدعم عنه من ميكروسوفت واصبح لا يدعم ميزات جافاسكربت الحديثة ورغم ذلك يتم استخدامه باعداد كبيرة جدا للذين مزالو يستخدمون اصدرات الويندوز القديم وهؤلاء يمثلون اكثر من اربعين بالمئة ورغم كل تحذيرات منظمة الويب العالمية

... مصدر

تاريخ رفع الدعم عن انترنت اكسبلورر في 17 أغسطس 2020

والافضل الا يستخدم بعد الان

متصفح عمو غوغل

ان متصفح غوغل كروم يعتمد على تقنية (الفى 8) وهى نفس التقنية التى يعتمد عليها سيرفر جافاسكربت المسماة (نود جى اس) وستلاحظ انه اسرع من فايرفوكس بمرحلة لان الفايرفوكس يعتمد على تقنية مختلفة لن نتطرق اليها الان ولكن لا تفرح بسرعة متصفحات (الفى 8) مثل غوغل كروم

(ومايكروزفت ادج) وكروميوم وغيرهم كثير جدا لان كل هذه السرعة على حساب الرمات والبيروسيور

الويبجكت

في جافاسكربت، الكائنات هي الملك. اهذه اللغة إذا كنت تفهم الكائنات، فأنت تفهم جافاسكربت.

في جافاسكربت، يعتبر "كل شيء" تقريبًا كائنًا

- إذا تم تعريفها باستخدام (يمكن أن تكون القيم المنطقية كائنات. (المحجوزة **new** الكلمة
- **new** إذا تم تعريفها باستخدام الكلمة) يمكن أن تكون الأرقام كائنات. (المحجوزة
- **new** إذا تم تعريفها باستخدام الكلمة) يمكن أن تكون النصوص كائنات. (المحجوزة
- التواريخ هي دائمًا كائنات
- الرياضيات هي دائمًا كائنات
- التعبيرات العادية هي دائمًا كائنات

- المصفوفات هي دائمًا كائنات
- الوظائف هي دائمًا كائنات
- الكائنات هي دائمًا كائنات

جميع قيم جافاسكربت، باستثناء الأوليات، هي كائنات

البدائيات

القيمة الأولية هي القيمة التي ليس لها خصائص أو طرق

هي قيمة بدائية 3.14

نوع الكلمات البدائي هو الكلمات التي لها قيمة بدائية

تحدد جافاسكربت 7 أنواع من أنواع الكلمات البدائية

أمثلة للأنواع

- string
- number
- boolean
- null
- undefined
- symbol
- bigint

غير قابل للتغيير

القيم الأولية غير قابلة للتغيير (فهي مشفرة ولا يمكن تغييرها)

لكن لا يمكنك تغيير قيمة 3.14، x فيمكنك تغيير قيمة، x = 3.14 إذا كانت

الكائنات هي المتغيرات

:يمكن أن تحتوي متغيرات جافاسكربت على قيم مفردة

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

.يمكن أن تحتوي متغيرات جافاسكربت أيضًا على العديد من القيم

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

تم كتابة قيم الكائنات على هيئة اسم: أزواج القيمة (الاسم والقيمة مفصولة بنقطتين)

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

كائن جافاسكربت عبارة عن مجموعة من القيم النصية

const من الممارسات الشائعة الإعلان عن الكائنات باستخدام الكلمة المحجوزة.

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

طرق الكائن

الأساليب هي الإجراءات التي يمكن تنفيذها على الكائنات.
يمكن أن تكون خصائص الكائن قيمًا أولية وكائنات ووظائف أخرى.
طريقة الكائن هي خاصية كائن تحتوي على تعريف دالة

كائنات جافاسكربت عبارة عن حاويات للقيم النصية، تسمى الخصائص والأساليب

سوف تتعلم المزيد عن الأساليب في الفصول التالية

إنشاء كائن

باستخدام جافاسكربت، يمكنك تحديد وإنشاء الكائنات

هناك طرق مختلفة لإنشاء كائنات جديدة

- قم بإنشاء كائن واحد باستخدام كائن حرفي

- **new** قم بإنشاء كائن واحد باستخدام الكلمة المحجوزة.
- قم بتعريف مُنشئ كائن، ثم قم بإنشاء كائنات من النوع المُنشأ.
- **Object.create()** قم بإنشاء كائن باستخدام.

باستخدام كائن حرفي

هذه هي أسهل طريقة لإنشاء كائن جافاسكربت

باستخدام كائن حرفي، يمكنك تحديد وإنشاء كائن في عبارة واحدة

الكائن الحرفي عبارة عن قائمة بأزواج الاسم: القيمة (مثل العمر: 50) داخل `{}` الأقواس المتعرجة

يقوم **المثال** التالي بإنشاء كائن جافاسكربت جديد بأربع خصائص

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث أبو 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 الثالث ابو حبيب الحسيني

يقوم هذا **المثال** بإنشاء كائن جافاسكربت فارغ، ثم يضيف 4 خصائص

مثال

```
const hAbiB = {};
```

```
hAbiB.firstName = "Habib";
```

```
hAbiB.lastName = "Al Husini 😊😊😊";
```

```
hAbiB.age = 50;
```

```
hAbiB.eyeColor = "blue";
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

باستخدام الكلمة المحبوزة الجديدة

، **new Object()** يقوم **المثال** التالي بإنشاء كائن جافاسكربت جديد باستخدام
ثم يضيف 4 خصائص

مثال

```
const hAbiB = new Object();
```

```
hAbiB.firstName = "Habib";
```

```
hAbiB.lastName = "Al Husini 😊😊😊";
```

```
hAbiB.age = 50;
```

```
hAbiB.eyeColor = "blue";
```

الأمثلة المذكورة أعلاه تفعل الشيء نفسه تمامًا.

`new Object()` ولكن ليست هناك حاجة للاستخدام

لسهولة القراءة والبساطة وسرعة التنفيذ، استخدم الطريقة الحرفية للكائن

كائنات قابلة للتغيير

الكائنات قابلة للتغيير: تتم معالجتها حسب المرجع، وليس حسب القيمة : إذا كان العنصر كائنًا، فلن تؤدي العبارة التالية إلى إنشاء نسخة من العنصر

```
const x = hAbiB; // Will not create a copy of hAbiB.
```

والعنصر هما نفس `x` ليس نسخة من العنصر. إنه شخص. كل من `x` الكائن الكائن.

والعنصر هما نفس `x` سوف تغير العنصر أيضًا، لأن `x` أي تغييرات على الكائن.

مثال

```
const hAbiB = {  
  firstName:"Habib",  
  lastName:"Al Husini 😊😊😊",  
  age:50, eyeColor:"blue"
```

```
| }
```

```
| const x = hAbiB;
```

```
| x.age = 10; // Will change both x.age and hAbiB.age
```

خصائص الكائن

الخصائص هي الجزء الأكثر أهمية في أي كائن جافاسكربت.

الخصائص

الخصائص هي القيم المرتبطة بكائن جافاسكربت.

كائن جافاسكربت عبارة عن مجموعة من الخصائص غير المرتبة.

يمكن عادةً تغيير الخصائص وإضافتها وحذفها، ولكن بعضها للقراءة فقط.

الوصول إلى الخصائص

بناء الجملة للوصول إلى خاصية الكائن هو:

```
objectName.property      </i> // hAbiB.age
```

أو

```
objectName["property"] // hAbiB["age"]
```

أو

```
objectName[expression] // x = "age"; hAbiB[x]
```

يجب أن يتم تقييم التعبير إلى اسم خاصية.

مثال 1

```
hAbiB.firstname + " is " + hAbiB.age + " years old.;"
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

مثال 2

```
hAbiB["firstname"] + " is " + hAbiB["age"] + " years  
old.;"
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

الحلقة فور أن

خلال خصائص الكائن **for...in** تتكرر عبارة جافاسكربت

بناء الجملة

```
for (let variable in object) {  
    // code to be executed  
}
```

مرة واحدة لكل خاصية **for...in** سيتم تنفيذ كتلة التعليمات البرمجية داخل الحلقة:
التكرار من خلال خصائص الكائن

مثال

```
const hAbiB = {  
    fname: "Habib",  
    lname: "Al Husini 😊😊😊",  
    age: 25  
};  
  
for (let x in hAbiB) {  
    hAbiB += hAbiB[x];  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

إضافة خصائص جديدة

يمكنك إضافة خصائص جديدة إلى كائن موجود ببساطة عن طريق إعطائه قيمة.

افتراض أن كائن العنصر موجود بالفعل - يمكنك بعد ذلك منحه خصائص جديدة:

مثال

```
hAbiB.nationality = "Arabic";
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

حذف الخصائص

:المحجوزة تحذف خاصية من كائن **delete** الكلمة

مثال

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

```
delete hAbiB.age;
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

أو حذف العنصر ["العمر"]؛

مثال

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

```
delete hAbiB["age"];
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

المحجوزة كلاً من قيمة الخاصية والخاصية نفسها **delete** تحذف الكلمة.

بعد الحذف، لا يمكن استخدام الخاصية قبل إضافتها مرة أخرى.

لاستخدامه في خصائص الكائن. ليس له أي تأثير على **delete** تم تصميم العامل المتغيرات أو الوظائف.

لا ينبغي استخدام عامل التشغيل في خصائص كائنات جافاسكربت **delete** المحددة مسبقاً. يمكن أن يتعطل التطبيق.

الكائنات المتداخلة

يمكن أن تكون القيم الموجودة في كائن كائناً آخر

مثال

```
abib = {  
  name: "Habib",
```

```
age:30,  
HaBiB: {  
  $_Habib1:"Mohamid*"  
  $_Habib2:"Abu Habib Al-Husini"  
  $_Habib3:"Mohamed"  
}  
}
```

يمكنك الوصول إلى الكائنات المتداخلة باستخدام علامة النقطة أو علامة القوس:

مثال

```
abib.HaBiB.$_Habib2;
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

أو:

مثال

```
abib.HaBiB["$_Habib2"];
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

أو:

مثال

```
abib ["HaBiB"]["$_Habib2"];
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

أو:

مثال

```
let p1 = "HaBiB";  
let p2 = "$_Habib2";  
abib [p1][p2];
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

المصفوفات والكائنات المتداخلة

يمكن أن تكون القيم في الكائنات مصفوفات، والقيم في المصفوفات يمكن أن تكون كائنات:

مثال

```
const abib = {  
  name: "Habib",  
  age: 30,  
  HaBiB: [  
    {name: "Mohamid*", models: ["Fiesta", "Focus", "Musta  
ng"]},  
    {name: "Abu Habib Al-Husini", models:  
["320", "Husin", "hamad"]},  
    {name: "Mohamed", models: ["20", "Panda"]}  
  ]  
}
```

لكل `for-in` للوصول إلى المصفوفات داخل المصفوفات، استخدم حلقة مصفوفة:

مثال

```
for (let i in abib.HaBiB) {  
  x += "<h1>" + abib.HaBiB[i].name + "</h1>";  
  for (let z in abib.HaBiB[i].models) {  
    x += abib.HaBiB[i].models[z];  
  }  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

سمات الخاصية

جميع الخصائص لها اسم. وبالإضافة إلى ذلك لديهم أيضا قيمة.

. القيمة هي إحدى سمات الخاصية

. السمات الأخرى هي: قابلة للتعداد، وقابلة للتكوين، وقابلة للكتابة

تحدد هذه السمات كيفية الوصول إلى الخاصية (هل هي قابلة للقراءة؟، هل هي قابلة للكتابة؟)

في جافاسكربت، يمكن قراءة جميع السمات، ولكن يمكن تغيير سمة القيمة فقط (و فقط إذا كانت الخاصية قابلة للكتابة).

(يحتوي ايكما 5 على طرق للحصول على جميع سمات الخاصية وتعيينها)

خصائص النموذج الأولي

تترث كائنات جافاسكربت خصائص النموذج الأولي الخاص بها

لا تحذف الكلمة المحجوزة الخصائص الموروثة، ولكن إذا قمت بحذف **delete** خاصية نموذج أولي، فسوف يؤثر ذلك على جميع الكائنات الموروثة من النموذج الأولي .

أساليب كائن

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

أنظر أيضا:

سلسلة المحترف فى جافا سكربت قريبا اقوى من
سلسلة الكافى ان شاء الله

الطرق

أساليب جافاسكربت هي إجراءات يمكن تنفيذها على الكائنات .
طريقة جافاسكربت هي خاصية تحتوي على تعريف دالة

الأساليب هي وظائف مخزنة كخصائص الكائن

الوصول إلى أساليب الكائن

:يمكنك الوصول إلى أسلوب كائن باستخدام بناء الجملة التالي

```
objectName.methodName()
```

كتابع لكائن العنصر ، والاسم الكامل كخاصية **fullName()** ستصف عادةً
عند استدعائها باستخدام (كدالة) **fullName** سيتم تنفيذ الخاصية
: لكائن العنصر **fullName()** يصل هذا **المثال** إلى طريقة

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

بدون ()، فسوف يُرجع تعريف `fullName` ، إذا قمت بالوصول إلى خاصية الوظيفة :

مثال

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

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

إضافة طريقة إلى كائن

من السهل إضافة طريقة جديدة إلى كائن

مثال

```
hAbiB.name = function () {  
    return this.firstName + " " + this.lastName;  
};
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

استخدام الطرق المضمنة

طريقة كائن النص لتحويل النص إلى `toUpperCase()` يستخدم هذا المثال
:أحرف كبيرة

```
let message = "Abu Habib Al Husini *_*!";
```

```
let x = message.toUpperCase();
```

بعد تنفيذ الكود أعلاه ستكون x قيمة

```
Abu Habib Al Husini *_*!
```

مثال

```
hAbiB.name = function () {  
  return (this.firstName + "  
  " + this.lastName).toUpperCase();  
};
```

عرض الكائنات

كيفية عرض الكائنات ؟

[object Object] سيؤدي عرض كائن جافاسكربت إلى إخراج

مثال

```
const hAbiB = {  
  name: "Habib",  
  age: 30,  
  city: "Hosini"  
};
```

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

بعض الحلول الشائعة لعرض كائنات جافاسكربت هي:

- عرض خصائص الكائن بالاسم
- عرض خصائص الكائن في حلقة
- `Object.values()` عرض الكائن باستخدام
- `JSON.stringify()` عرض الكائن باستخدام

عرض خصائص الكائن

: يمكن عرض خصائص الكائن كنص

مثال

```
const hAbiB = {  
  name: "Habib",  
  age: 30,  
  city: "Hosini"
```

```
};
```

```
document.getElementById("Habib").innerHTML =  
hAbiB.name + "," + hAbiB.age + "," + hAbiB.city;
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

عرض الكائن في حلقة

يمكن جمع خصائص الكائن في حلقة:

مثال

```
const hAbiB = {  
  name: "Habib",  
  age: 30,  
  city: "Hosini"  
};
```

```
let hAbiB = "";  
for (let x in hAbiB) {  
  hAbiB += hAbiB[x] + " ";  
};
```

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

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

في الحلقة [x] يجب عليك استخدام العنصر.

(متغير x لأن hAbiB.x لن يعمل).

Object.values() استخدام

يمكن تحويل أي كائن جافاسكربت إلى مصفوفة باستخدام **Object.values()**:

```
const hAbiB = {  
  name: "Habib",  
  age: 30,  
  city: "Hosini"  
};
```

```
const Habib_Array = Object.values(hAbiB);
```

أصبح الآن مصفوفة جافاسكربت، جاهزة للعرض **Habib_Array**:

مثال

```
const hAbiB = {  
  name: "Habib",  
  age: 30,  
  city: "Hosini"  
};
```

```
const Habib_Array = Object.values(hAbiB);
```

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

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

JSON.stringify() باستخدام

يمكن تحويل أي كائن جافاسكربت إلى نص (تحويله إلى نص) باستخدام وظيفة **JSON.stringify()**: جافاسكربت

```
const hAbiB = {  
  name: "Habib",  
  age: 30,  
  city: "Hosini"  
};
```

```
let myString = JSON.stringify(hAbiB);
```

أصبحت الآن نص جافاسكربت، جاهزة للعرض **myString**:

مثال

```
const hAbiB = {  
  name: "Habib",  
  age: 30,  
  city: "Hosini"  
};
```

```
let myString = JSON.stringify(hAbiB);
document.getElementById("Habib").innerHTML =
myString;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

JSON ستكون النتيجة نص تتبع تدوين

```
{"اسم": "حبيب", "العمر": 50, "المدينة": "نيويورك"}
```

تسلسل التواريخ

تحويل التواريخ إلى نصوص `JSON.stringify`:

مثال

```
const hAbiB = {
  name: "Habib",
  today: new Date()
};
```

```
let myString = JSON.stringify(hAbiB);
document.getElementById("Habib").innerHTML =
myString;
```

Stringify وظائف

لن يقيّد الوظائف `JSON.stringify`

مثال

```
const hAbiB = {  
  name: "Habib",  
  age: function () {return 30;}  
};
```

```
let myString = JSON.stringify(hAbiB);  
document.getElementById("Habib").innerHTML =  
myString;
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

يمكن "إصلاح" هذا إذا قمت بتحويل الوظائف إلى نصوص قبل التوثيق

مثال

```
const hAbiB = {  
  name: "Habib",  
  age: function () {return 30;}  
};  
hAbiB.age = hAbiB.age.toString();
```

```
let myString = JSON.stringify(hAbiB);
document.getElementById("Habib").innerHTML =
myString;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تصنيف المصفوفات

من الممكن أيضاً تقييد مصفوفات جافاسكربت

مثال

```
const arr = ["Habib", "Habib", "Sally", "Hosini"];
```

```
let myString = JSON.stringify(arr);
document.getElementById("Habib").innerHTML =
myString;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

JSON ستكون النتيجة نص تتبع تدوين

```
["حبيب", "حسيني", "سالي", "حسيني"]
```

ملفات الكائنات

ملفات (Getters وSetters)

Getters وSetters برنامجي (ES5 2009) قدم ايكما 5

تحديد أدوات الوصول إلى الكائنات Getters وSetters تتيح لك أدوات (الخصائص المحسوبة).

(الحصول على الكلمة المحبوزة) Getter

language لقيمة الخاصية **get** خاصية **lang** يستخدم هذا المثال

مثال

```
// Create an object:
const hAbiB = {
  firstName: "Habib",
  lastName: "Al Husini 😊😊😊",
  language: "en",
  get lang() {
    return this.language;
  }
};
```

```
// Display data from the object using a getter:  
document.getElementById("Habib").innerHTML =  
hAbiB.lang;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الوصول السريع للبيانات

language لقيمة الخاصية **set** خاصية **lang** يستخدم هذا المثال

مثال

```
const hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  language: "",  
  set lang(lang) {  
    this.language = lang;  
  }  
};
```

```
// Set an object property using a setter:  
hAbiB.lang = "en";
```

```
// Display data from the object:  
document.getElementById("Habib").innerHTML =  
hAbiB.language;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تابع جلب البيانات

ما هو الفرق بين هذين المثالين؟

1 مثال

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

```
// Display data from the object using a method:  
document.getElementById("Habib").innerHTML =  
hAbiB.fullName();
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

2 مثال

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



```
};
```

```
// Display data from the object using a getter:  
document.getElementById("Habib").innerHTML =  
hAbiB.fullName;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

hAbiB.fullName(): المثال 1: الوصول إلى الاسم الكامل كدالة
hAbiB.fullName: المثال 2: الوصول إلى الاسم الكامل كخاصية
يوفر المثال الثاني بناء جملة أبسط.

جودة الكلمات

يمكن لـ جافاسكربت تأمين جودة أفضل للكلمات عند استخدام الحروف
والمحددات

language الخاصة، في هذا المثال، يتم إرجاع قيمة الخاصية lang باستخدام
بالأحرف الكبيرة:

مثال

```
// Create an object:  
const hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 🇸🇦🇸🇦🇸🇦",  
  language: "en",  
  get lang() {  
    return this.language.toUpperCase();  
  }  
};
```

```
}  
};
```

// Display data from the object using a getter:

```
document.getElementById("Habib").innerHTML =  
hAbiB.lang;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

language الخاصية، في هذا المثال، يتم تخزين قيمة كبيرة في lang باستخدام الخاصية:

مثال

```
const hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  language: "Arabic",  
  set lang(lang) {  
    this.language = lang.toUpperCase();  
  }  
};
```

// Set an object property using a setter:

```
hAbiB.lang = "en";
```

// Display data from the object:

```
document.getElementById("Habib").innerHTML =  
hAbiB.language;
```

Getters و Setters لماذا استخدام

- أنه يعطي بناء الجملة أبسط
- يسمح ببناء جملة متساوٍ للخصائص والأساليب
- يمكنه تأمين جودة أفضل للكلمات
- إنه مفيد للقيام بالأشياء خلف الكواليس

Object.defineProperty()

لإضافة **Object.defineProperty()** يمكن أيضاً استخدام الطريقة **Getters و Setters:**

مثال مضاد

```
// Define object
```

```
const obj = {counter : 0};
```

```
// Define setters and getters
```

```
Object.defineProperty(obj, "reset", {  
  get : function () {this.counter = 0;}  
});
```

```
Object.defineProperty(obj, "increment", {  
  get : function () {this.counter++;}  
});
```

```
Object.defineProperty(obj, "decrement", {
```

```
get : function () {this.counter--;}  
});  
Object.defineProperty(obj, "add", {  
  set : function (value) {this.counter += value;}  
});  
Object.defineProperty(obj, "subtract", {  
  set : function (value) {this.counter -= value;}  
});
```

```
// Play with the counter:
```

```
obj.reset;  
obj.add = 5;  
obj.subtract = 1;  
obj.increment;  
obj.decrement;
```

منشئي الكائنات

مثال

```
function hAbiB(first, last, age, eye) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eye;  
}
```

ملحوظات

من الاستخدام الجيد تسمية وظائف المنشئ بحرف أول كبير.

أنواع الكائنات (المخططات) (الفئات)

والأمثلة من الفصول السابقة محدودة. إنهم ينشئون كائنات فردية فقط في بعض الأحيان نحتاج إلى "مخطط" لإنشاء العديد من الكائنات من نفس النوع.

. طريقة إنشاء "نوع الكائن" هي استخدام وظيفة منشئ الكائن

.هي وظيفة منشئ الكائن **function hAbiB()**، في المثال أعلاه

يتم إنشاء الكائنات من نفس النوع عن طريق استدعاء وظيفة المنشئ:
المحجوزة **new** بالكلمة

```
const myFather = new hAbiB("Habib", "Al Husini 🇸🇩🇸🇩",  
🇸🇩", 50, "blue");  
const myMother  
= new hAbiB("Sally", "Rally", 48, "green");
```

إضافة خاصية إلى كائن

من السهل إضافة خاصية جديدة إلى كائن موجود

مثال

```
myFather.nationality = "Arabic";
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

(وليس لأي كائنات شخص آخر). . myFather ستتم إضافة الخاصية إلى

إضافة طريقة إلى كائن

من السهل إضافة طريقة جديدة إلى كائن موجود

مثال

```
myFather.name = function () {  
    return this.firstName + " " + this.lastName;  
};
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

ليس لأمي. (وليس لأي كائنات شخص myFather. سيتم إضافة الطريقة إلى
آخر)

إضافة خاصية إلى منشئ

لا يمكنك إضافة خاصية جديدة إلى منشئ الكائن بنفس الطريقة التي تضيف بها خاصية جديدة إلى كائن موجود:

مثال

```
hAbiB.nationality = "Arabic";
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

لإضافة خاصية جديدة إلى المنشئ، يجب عليك إضافتها إلى وظيفة المنشئ

مثال

```
function hAbiB(first, last, age, eyeColor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyeColor;  
  this.nationality = "Arabic";  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

بهذه الطريقة يمكن أن يكون لخصائص الكائن قيم افتراضية

إضافة طريقة إلى منشئ

يمكن لوظيفة المنشئ أيضًا تحديد الطرق

مثال

```
function hAbiB(first, last, age, eyeColor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyeColor;  
  this.name = function() {  
    return this.firstName + " " + this.lastName;  
  };  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

لا يمكنك إضافة أسلوب جديد إلى منشئ الكائن بنفس الطريقة التي تضيف بها أسلوبًا جديدًا إلى كائن موجود.

يجب أن تتم إضافة الأساليب إلى منشئ الكائن داخل وظيفة المنشئ

مثال

```
function hAbiB(firstName, lastName, age, eyeColor) {  
  this.firstName = firstName;  
  this.lastName = lastName;  
  this.age = age;  
  this.eyeColor = eyeColor;  
}
```



```
this.changeName = function (name) {  
  this.lastName = name;  
};  
}
```

بتعيين قيمة الاسم لخاصية الاسم الأخير ChangeName() تقوم الدالة للشخص.

الآن يمكنك تجربة:

```
myMother.changeName("Al Husini 😊😊😊");
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تعرف جافاسكربت العنصر الذي تتحدث عنه عن طريق myMother "استبدال" هذا بـ

كائنات جاهزة في اللغة

:تحتوي جافاسكربت على مُنشآت مضمنة للكائنات الأصلية

```
new String() // A new String object  
new Number() // A new Number object  
new Boolean() // A new Boolean object  
new Object() // A new Object object  
new Array() // A new Array object  
new RegExp() // A new RegExp object  
new Function() // A new Function object  
new Date() // A new Date object
```

لا يمكن **new** هو كائن عام **Math**. غير موجود في القائمة **Math()** الكائن **Math**. استخدام الكلمة المحجوزة في

هل كنت تعلم؟

كما ترون أعلاه، تحتوي جافاسكربت على إصدارات كائنات من أنواع الكلمات ولكن لا يوجد سبب لإنشاء كائنات **Boolean** و **Number** و **String** الأولية معقدة. القيم البدائية أسرع بكثير

new String() استخدم نص حرفية **""** بدلاً من

new Number() استخدم الأرقام الحرفية **50** بدلاً من

new Boolean() بدلاً من **true / false** استخدم القيم الحرفية المنطقية

new Object() استخدم القيم الحرفية للكائن **{}** بدلاً من

new Array() استخدم المصفوفة الحرفية **[]** بدلاً من

new RegExp() استخدم الخاصية الحرفي **/()/** بدلاً من

new Function() استخدم التعبيرات الوظيفية **()** بدلاً من

مثال

```
| let x1 = "";           // new primitive string
| let x2 = 0;           // new primitive number
| let Husin = false;   // new primitive boolean
```

```
const x4 = {}; // new Object object
const hamad = []; // new Array object
const x6 = /()/ // new RegExp object
const x7 = function(){}; // new function
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

كائنات النص

`firstName = "Habib"`: عادة، يتم إنشاء النصوص كعناصر أولية المحجوزة `new` ولكن يمكن أيضًا إنشاء النصوص ككائنات باستخدام الكلمة `firstName = new String("Habib")`.
JS تعرف على سبب عدم إنشاء النصوص ككائن في الفصل نصوص

كائنات الأرقام

`x = 30`: عادة، يتم إنشاء الأرقام كأعداد أولية الكلمة المحجوزة `new` ولكن يمكن أيضًا إنشاء الأرقام ككائنات باستخدام `x = new Number(30)`.
JS تعرف على سبب عدم إنشاء الأرقام ككائن في الفصل أرقام

الكائنات المنطقية

`x = false`: عادة، يتم إنشاء القيم المنطقية كأوليات

:المحجوزة **new** ولكن يمكن أيضاً إنشاء القيم المنطقية ككائنات باستخدام الكلمة
x = new Boolean(false)

. **JS Booleans** تعرف على سبب عدم إنشاء القيم المنطقية ككائن في الفصل

نماذج كائنات

ترث كافة كائنات جافاسكربت الخصائص والأساليب من النموذج الأولي

: تعلمنا في الفصل السابق كيفية استخدام مُنشئ الكائن

مثال

```
function hAbiB(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
}
```

```
const myFather = new hAbiB("Habib", "Al Husini 😊😊  
😊", 50, "blue");  
const myMother  
= new hAbiB("Sally", "Rally", 48, "green");
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

:لقد تعلمنا أيضًا أنه لا يمكنك إضافة خاصية جديدة إلى مُنشئ كائن موجود

مثال

```
hAbiB.nationality = "Arabic";
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

:لإضافة خاصية جديدة إلى المنشئ، يجب عليك إضافتها إلى وظيفة المنشئ

مثال

```
function hAbiB(first, last, age, eyeColor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyeColor;  
  this.nationality = "Arabic";  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

وراثة النموذج الأولي

:ترث جميع كائنات جافاسكربت الخصائص والأساليب من النموذج الأولي

- **Date.prototype** الكائنات ترث من **Date**
- **Array.prototype** الكائنات ترث من **Array**
- **hAbiB.prototype** الكائنات ترث من **hAbiB**

في الجزء العلوي من نص الوراثة النمذجية **Object.prototype** يوجد ترث **hAbiB** والكائنات والكائنات **Array** الكائنات **Date** من **Object.prototype**.

إضافة خصائص وأساليب للكائنات

في بعض الأحيان تريد إضافة خصائص (أو أساليب) جديدة إلى كافة الكائنات الموجودة من نوع معين.
في بعض الأحيان تريد إضافة خصائص (أو أساليب) جديدة إلى مُنشئ الكائن.

باستخدام خاصية النموذج الأولي

بإضافة خصائص جديدة لمنشئ **prototype** تسمح لك خاصية جافاسكربت الكائنات:

مثال

```
function hAbiB(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
}
```

```
hAbiB.prototype.nationality = "Arabic";
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

أيضًا إضافة أساليب جديدة لمنشئي **prototype** تتيح لك خاصية جافاسكربت الكائنات:

مثال

```
function hAbiB(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
}  
  
hAbiB.prototype.name = function() {  
  return this.firstName + " " + this.lastName;  
};
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

قم بتعديل النماذج الأولية فقط . لا تتم مطلقاً بتعديل النماذج الأولية لكائنات جافاسكربت القياسية.

عناصر التكرارية

for..of. الكائنات القابلة للتكرار هي كائنات يمكن تكرارها باستخدام

Symbol.iterator من الناحية الفنية، يجب على العناصر التكرارية تنفيذ هذه الطريقة.

التكرار على نص

: حلقة للتكرار على عناصر النص **for..of** يمكنك استخدام

مثال

```
for (const x of "Abu Habib alhosiny") {  
  // code block to be executed  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

التكرار على مصفوفة

: حلقة للتكرار على عناصر المصفوفة **for..of** يمكنك استخدام

مثال

```
for (const x of [1,2,3,4,5]) {  
  // code block to be executed  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

استخدام نكست

يحدد بروتوكول التكرار كيفية إنتاج نص من القيم من كائن

طريقة ما **next()** يصبح الكائن مكرراً عندما ينفذ

بإرجاع كائن له خاصيتين **next()** يجب أن تقوم الطريقة

- القيمة (القيمة التالية)
- تم (صحيح أو خطأ)

طريقة نكست والإرجاع المتعدد

يتم استدعاء كل **next()**.... هذه العوائد التكرارية لا تنتهي أبداً: 10,20,30,40
مرة:

مثال

```
// Home Made Iterable
function Habib_Num() {
  let n = 0;
  return {
    next: function() {
      n += 10;
      return {value:n, done:false};
    }
  };
}
```

```
// Create Iterable
```

```
const n = Habib_Num();
n.next(); // Returns 10
n.next(); // Returns 20
n.next(); // Returns 30
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

أصبحت مشكلة المنزل قابلة للتكرار

عبرة جافاسكربت **for..of** لا يدعم

Code.iterator كائن جافاسكربت القابل للتكرار هو كائن يحتوي على

دالة **next()** دالة تقوم بإرجاع **Symbol.iterator** هي

for (const x of iterable) { } يمكن تكرار التكرار باستخدام الكود

مثال

```
// Create an Object
```

```
Habib_Num = {};
```

```
// Make it Iterable
```

```
Habib_Num[Symbol.iterator] = function() {
```

```
  let n = 0;
```

```
  done = false;
```

```
  return {
```

```
    next() {
```

```
      n += 10;
```

```
      if (n == 100) {done = true}
```

```
return {value:n, done:done};  
}  
};  
}
```

for..of الآن يمكنك استخدام

```
for (const num of Habib_Num) {  
  // Any Code Here  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

for..of تلقائيًا بواسطة **Sympo.iterator** يُستدعى التابع
:"ولكن يمكننا أيضًا القيام بذلك" يدويًا

مثال

```
let iterator = Habib_Num[Symbol.iterator]();  
  
while (true) {  
  const result = iterator.next();  
  if (result.done) break;  
  // Any Code Here  
}
```

مجموعات

مجموعة جافاسكربت عبارة عن مجموعة من القيم الفريدة

يمكن أن تحدث كل قيمة مرة واحدة فقط في المجموعة

يمكن أن تحتوي المجموعة على أي قيمة لأي نوع كلمات

تعيين الأساليب

Method	وصف
new Set()	إنشاء مجموعة جديدة
add()	يضيف عنصرًا جديدًا إلى المجموعة
delete()	إزالة عنصر من مجموعة
has()	يُرجع صحيحًا في حالة وجود قيمة
clear()	يزيل كافة العناصر من مجموعة
forEach()	يستدعي رد اتصال لكل عنصر
values()	إرجاع حلقة بجميع القيم الموجودة في المجموعة
keys()	جلب المفاتيح
entries()	إرجاع حلقة مع أزواج [القيمة، المفتاح] من مجموعة

Property وصف

size إرجاع العناصر العددية في المجموعة

كيفية إنشاء مجموعة

يمكنك إنشاء مجموعة جافاسكربت عن طريق

- `new Set()` تمرير مصفوفة إلى
- لإضافة القيم `add()` قم بإنشاء مجموعة جديدة واستخدمها
- لإضافة المتغيرات `add()` قم بإنشاء مجموعة جديدة واستخدمها

طريقة Set ()

:المنشئ `new Set()` تمرير مصفوفة إلى

مثال

```
// Create a Set  
const Hbib = new Set(["a", "b", "c"]);
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

:قم بإنشاء مجموعة وإضافة قيم حرفية

مثال

```
// Create a Set  
const Hbib = new Set();
```

```
// Add Values to the Set
```

```
Hbib.add("a");  
Hbib.add("b");  
Hbib.add("c");
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

إنشاء مجموعة وإضافة المتغيرات

مثال

```
// Create Variables  
const a = "a";  
const b = "b";  
const c = "c";  
  
// Create a Set  
const Hbib = new Set();  
  
// Add Variables to the Set  
Hbib.add(a);  
Hbib.add(b);  
Hbib.add(c);
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

الإضافة

مثال

```
Hbib.add("d");  
Hbib.add("e");
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

:إذا أضفت عناصر متساوية، فسيتم حفظ العنصر الأول فقط

مثال

```
Hbib.add("a");  
Hbib.add("b");  
Hbib.add("c");  
Hbib.add("c");  
Hbib.add("c");  
Hbib.add("c");  
Hbib.add("c");  
Hbib.add("c");
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

طريقة forEach()

Set دالة لكل عنصر **forEach()** تستدعي الطريقة

مثال

```
// Create a Set
const Hbib = new Set(["a","b","c"]);

// List all entries
let BiB = "";
Hbib.forEach (function(value) {
BiB += value;
})
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

القيم

الذي يحتوي على جميع القيم **Iterator** كائن **values()** تُرجع الطريقة
الموجودة في المجموعة:

مثال

```
Hbib.values() // Returns [object Set Iterator]
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

:للوصول إلى العناصر **Iterator** يمكنك الآن استخدام كائن

مثال

```
// Create an Iterator
const myIterator = Hbib.values();

// List all Values
let BiB = "";
for (const entry of myIterator) {
  BiB += entry;
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

المفاتيح

المجموعة لا تحتوي على مفاتيح.

values() ترجع نفس **keys()**.

وهذا يجعل المجموعات متوافقة مع الخرائط

مثال

```
Hbib.keys() // Returns [object Set Iterator]
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

. الإدخالات

المجموعة لا تحتوي على مفاتيح.

تُرجع أزواج [القيمة، القيمة] بدلاً من أزواج [المفتاح، القيمة] **entries()**

وهذا يجعل المجموعات متوافقة مع الخرائط

مثال

```
// Create an Iterator
const myIterator = Hbib.entries();

// List all Entries
let BiB = "";
for (const entry of myIterator) {
  BiB += entry;
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

المجموعات هي كائنات

يتم إرجاع الكائن **typeof**، بالنسبة للمجموعة

| **typeof Hbib; // Returns object**

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

يتم إرجاع صحيح **instanceof Set**، بالنسبة للمجموعة

| **Hbib instanceof Set; // Returns true**

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

تغيير قيمة الخاصية

بناء الجملة

| **Object.defineProperty(object, property, {value : value})**

هذا **المثال** يغير قيمة الخاصية

مثال

```
const hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  language: "EN"  
};
```

```
// Change a property
```

```
Object.defineProperty(hAbiB, "language", {value: "NO"  
});
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تغيير كلمات التعريف

بتغيير الكلمات التعريفية للخاصية التالية ES5 يسمح

```
writable: true // Property value can be changed  
enumerable: true // Property can be enumerated  
configurable: true // Property can be reconfigured
```

```
writable: false // Property value can not be changed  
enumerable: false // Property can be not enumerated  
configurable: false // Property can be not reconfigured
```

بتغيير الحروف والمحددات ES5 يسمح

```
// Defining a getter
```

```
get: function() { return language }
```

```
// Defining a setter
```

```
set: function(value) { language = value }
```

هذا المثال يجعل اللغة للقراءة فقط:

```
Object.defineProperty(hAbiB, "language",  
{writable:false});
```

هذا المثال يجعل اللغة غير قابلة للإحصاء:

```
Object.defineProperty(hAbiB, "language",  
{enumerable:false});
```

قائمة جميع الخصائص

يسرد هذا المثال جميع خصائص الكائن:

مثال

```
const hAbiB = {  
  firstName: "Habib",  
  lastName : "Al Husini 😊😊😊",  
  language : "EN"  
};
```

```
Object.defineProperty(hAbiB, "language",  
{enumerable:false});
```

`Object.getOwnPropertyNames(hAbiB); // Returns an array of properties`

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

قائمة خصائص لا تعد ولا تحصى

:يسرد هذا المثال فقط الخصائص القابلة للإحصاء للكائن

مثال

```
const hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  language: "EN"  
};
```

```
Object.defineProperty(hAbiB, "language",  
{enumerable: false});  
Object.keys(hAbiB); // Returns an array of enumerable  
properties
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

إضافة

:يضيف هذا المثال خاصية جديدة إلى كائن

الكافي في جافاسكريبت الجزء Abu Habib Al-Husini الصفحة 166

الثالث

مثال

```
// Create an object:
```

```
const hAbiB = {  
  firstName: "Habib",  
  lastName: "Al Husini 😊😊😊",  
  language: "EN"  
};
```

```
// Add a property
```

```
Object.defineProperty(hAbiB, "year", {value:"2008"});
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

Getters و Setters إضافة

لإضافة `Object.defineProperty()` يمكن أيضًا استخدام الطريقة
Getters و Setters:

مثال

```
//Create an object
```

```
const hAbiB = {firstName:"Habib", lastName:"Al  
Husini 😊😊😊"};
```

```
// Define a getter
```

```
Object.defineProperty(hAbiB, "fullName", {  
  get: function () {return this.firstName + "
```

```
" + this.lastName;}  
});
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

مثال

مثال

```
// Define object  
const obj = {counter:0};  
  
// Define setters  
Object.defineProperty(obj, "reset", {  
  get : function () {this.counter = 0;}  
});  
Object.defineProperty(obj, "increment", {  
  get : function () {this.counter++;}  
});  
Object.defineProperty(obj, "decrement", {  
  get : function () {this.counter--;}  
});  
Object.defineProperty(obj, "add", {  
  set : function (value) {this.counter += value;}  
});  
Object.defineProperty(obj, "subtract", {  
  set : function (i) {this.counter -= i;}  
});
```



```
// Play with the counter:
```

```
obj.reset;
```

```
obj.add = 5;
```

```
obj.subtract = 1;
```

```
obj.increment;
```

```
obj.decrement;
```

بيانات الوظائف

الكلمة المحجوزة **function** يتم تعريف وظائف جافاسكربت باستخدام
يمكنك استخدام إعلان دالة أو تعبير دالة.

إعلانات الوظيفة

في وقت سابق من هذا الكتاب، تعلمت أنه يتم الإعلان عن الوظائف بالصيغة
التالية:

```
function functionName(parameters) {
```

```
// code to be executed
```

```
}
```

لا يتم تنفيذ الوظائف المعلنة على الفور. يتم "حفظها لاستخدامها باذن الله تعالى". وسيتنفيذها باذن الله تعالى ، عند استدعائها (استدعائها)

مثال

```
function Husini(a, b) {  
  return a * b;  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تُستخدم الفواصل المنقوطة لفصل عبارات جافاسكربت القابلة للتنفيذ. نظرًا لأن إعلان الوظيفة ليس بيانًا قابلاً للتنفيذ، فليس من الشائع إنهاءه بفاصلة منقوطة.

التعبيرات الوظيفية

. يمكن أيضًا تعريف وظيفة جافاسكربت باستخدام تعبير
:يمكن تخزين تعبير الدالة في متغير

مثال

```
const x = function (a, b) {return a * b};
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

:بعد تخزين تعبير دالة في متغير، يمكن استخدام المتغير كدالة

مثال

```
const x = function (a, b) {return a * b};  
let z = x(4, 3);
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

الوظيفة أعلاه هي في الواقع وظيفة مجهولة (وظيفة بدون اسم).

الوظائف المخزنة في المتغيرات لا تحتاج إلى أسماء الوظائف. يتم استدعاؤها دائماً (استدعاءها) باستخدام اسم المتغير.

تنتهي الوظيفة أعلاه بفاصلة منقوطة لأنها جزء من عبارة قابلة للتنفيذ.

منشئ الوظيفة

كما رأيت في الأمثلة السابقة، يتم تعريف وظائف جافاسكريبت باستخدام المحجوزة **function** الكلمة.

يمكن أيضاً تعريف الوظائف باستخدام منشئ وظائف جافاسكريبت مدمج **Function()**. يُسمى

مثال

```
const Husini = new Function("a", "b", "return a * b");
```

```
let x = Husini(4, 3);
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

:ليس عليك في الواقع استخدام مُنشئ الوظيفة. **المثال** أعلاه هو نفس كتابة

مثال

```
| const Husini = function (a, b) {return a * b};
```

```
| let x = Husini(4, 3);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الكلمة المحجوزة في **new** في أغلب الأحيان، يمكنك تجنب استخدام جافاسكربت.

وظيفة الرفع

في وقت سابق من هذا الكتاب، تعلمت عن "الرفع الرفع هو السلوك الافتراضي لجافاسكربت لنقل الإعلانات عن المتغيرات إلى أعلى النطاق الحالي. ينطبق الرفع على الإعلانات المتغيرة وعلى إعلانات الوظائف: ولهذا السبب، يمكن استدعاء وظائف جافاسكربت قبل الإعلان عنها

```
Husini(5);
```

```
function Husini(y) {
```

```
return y * y;
```

```
}
```

لا يتم رفع الوظائف المحددة باستخدام التعبير

وظائف الاستدعاء الذاتي

"يمكن جعل التعبيرات الوظيفية "استدعاء ذاتي".
يتم استدعاء (بدء) تعبير الاستدعاء الذاتي تلقائيًا، دون استدعائه
(). سيتم تنفيذ تعبيرات الدالة تلقائيًا إذا كان التعبير متبوعًا بـ
لا يمكنك استدعاء إعلان دالة ذاتيًا
:يجب عليك إضافة قوسين حول الدالة للإشارة إلى أنها تعبير دالة

مثال

```
(function () {  
  let x = "Abo Habib Al Hosini 😊😊!!"; // I will invoke  
  myself  
})();
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الوظيفة أعلاه هي في الواقع وظيفة استدعاء ذاتي مجهولة (وظيفة بدون اسم)

استخدام الوظائف كقيم

يمكن استخدام وظائف جافاسكربت كقيم

مثال

```
function Husini(a, b) {  
  return a * b;  
}
```

```
let x = Husini(4, 3);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

يمكن استخدام وظائف جافاسكربت في التعبيرات

مثال

```
function Husini(a, b) {  
  return a * b;  
}
```

```
let x = Husini(4, 3) * 2;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الوظائف هي كائنات

لوظائف "function" التشغيل في جافاسكربت بإرجاع **typeof** يقوم عامل

ولكن أفضل وصف لوظائف جافاسكربت هو أنها كائنات

• وظائف جافاسكربت لها كل من الخصائص والأساليب

بإرجاع عدد البرامتر المستلمة عند **arguments.length** تقوم الخاصية
:استدعاء الدالة

مثال

```
function Husini(a, b) {  
  return arguments.length;  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

: بإرجاع الوظيفة كنص **toString()** تقوم الطريقة

مثال

```
function Husini(a, b) {  
  return a * b;  
}
```

```
let BiB = Husini.toString();
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

تسمى الوظيفة التي يتم تعريفها على أنها خاصية لكائن ما طريقة للكائن

تسمى الوظيفة المصممة لإنشاء كائنات جديدة بمنشئ الكائن

الوظائف المختصرة

تسمح الوظائف المختصرة بتركيب جملة قصيرة لكتابة تعبيرات الوظائف المحجوزة، والأقواس **return** الكلمة المحجوزة، والكلمة **function** لا تحتاج إلى المتعرجة .

مثال

```
// ES5  
var x = function(x, y) {  
  return x * y;  
}
```

```
// ES6  
const x = (x, y) => x * y;
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

وهي ليست مناسبة تمامًا. **this** وظائف مختصرة ليس لها وظائفها الخاصة لتحديد أساليب الكائنات.

لا يتم رفع وظائف مختصرة. ويجب تعريفها قبل استخدامها.

لأن تعبير الدالة يكون دائمًا **var** أكثر أمانًا من الاستخدام **const** يعد الاستخدام قيمة ثابتة.

الكلمة المحجوزة والأقواس المتعرجة فقط إذا كانت الدالة **return** يمكنك حذف عبارة واحدة. ولهذا السبب، قد يكون من العادات الجيدة الاحتفاظ بها دائمًا

مثال

```
const x = (x, y) => { return x * y };
```


أو إصدار سابق IE11 وظائف مختصرة غير مدعومة في

معلومات الوظيفة

لا يقوم كود جافاسكربت بإجراء أي فحص لقيم البرمترات (الوسائط) **function**

معلومات الوظيفة او البرمترات

في وقت سابق من هذا الكتاب، تعلمت أن الوظائف يمكن أن تحتوي على معلومات :

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

معلومات الوظيفة هي الأسماء المدرجة في تعريف الوظيفة

وسيطات الدالة هي القيم الحقيقية التي تم تمريرها إلى الدالة (واستقبالها بواسطة)

قواعد البرامتر

لا تحدد بيانات وظائف جافاسكربت أنواع الكلمات للمعلومات

لا تقوم وظائف جافاسكربت بإجراء فحص النوع على البرامتر التي تم تمريرها.
لا تتحقق وظائف جافاسكربت من عدد البرامتر المستلمة.

البرمترات الافتراضية

إذا تم استدعاء دالة باستخدام وسيطات مفقودة (أقل من المعلن عنها)، فسيتم **undefined** تعيين القيم المفقودة إلى

في بعض الأحيان يكون هذا مقبولاً، ولكن في بعض الأحيان يكون من الأفضل :
تعيين قيمة افتراضية للبرامتر

مثال

```
function Husini(x, y) {  
  if (y === undefined) {  
    y = 2;  
  }  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو جيب الحسيني

قيم البرمترات الافتراضية

مثال

$y = 10$ أو لم يتم تعريفه، فإن y إذا لم يتم تمرير

```
function Husini(x, y = 10) {  
  return x + y;  
}  
Husini(5);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

البرامتر الغير محدود العدد

تسمح البرامتر الباقية (...) للدالة بمعاملة عدد غير محدد من البرامتر كمصفوفة:

مثال

```
function sum(...args) {  
  let sum = 0;  
  for (let arg of args) sum += arg;  
  return sum;  
}
```

```
let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

كائن البرمترات

. تحتوي وظائف جافاسكربت على كائن مضمن يسمى كائن البرامتر

يحتوي كائن الوسيطة على مصفوفة من البرامتر المستخدمة عند استدعاء الوظيفة (استدعاءها).

بهذه الطريقة يمكنك ببساطة استخدام دالة للعثور (على سبيل المثال) على أعلى قيمة في قائمة الأرقام

مثال

```
x = findMax(1, 123, 500, 115, 44, 88);
```

```
function findMax() {  
  let max = -Infinity;  
  for (let i = 0; i < arguments.length; i++) {  
    if (arguments[i] > max) {  
      max = arguments[i];  
    }  
  }  
  return max;  
}
```

كتاب الكافي في جافاسكربت الجزء الثالث أبو حبيب الحسيني

أو قم بإنشاء دالة لجمع كل قيم الإدخال

مثال

```
x = sumAll(1, 123, 500, 115, 44, 88);
```

```
function sumAll() {  
  let sum = 0;  
  for (let i = 0; i < arguments.length; i++) {
```

```
sum += arguments[i];  
}  
return sum;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثالث ابو حبيب الحسيني

إذا تم استدعاء دالة باستخدام عدد كبير جدًا من البرامتر (أكثر من المعلن عنها)،
. فيمكن الوصول إلى هذه البرامتر باستخدام كائن البرامتر

تمرير البرامتر حسب القيمة

.البرمترات، في استدعاء دالة، هي وسيطات الدالة
يتم تمرير وسيطات جافاسكريبت حسب القيمة : تتعرف الوظيفة فقط على القيم،
وليس مواقع الوسيطة
. إذا قامت دالة بتغيير قيمة الوسيطة، فإنها لا تغير القيمة الأصلية للبرامتر
.التغييرات التي يتم إجراؤها على البرامتر غير مرئية (تنعكس) خارج الوظيفة

تمرير الكائنات حسب المرجع

.في جافاسكريبت، مراجع الكائنات هي قيم
نوسبب هذا، سوف تتصرف الكائنات كما لو تم تمريرها حسب المرجع
.إذا قامت دالة بتغيير خاصية كائن، فإنها تغير القيمة الأصلية
تكون التغييرات في خصائص الكائن مرئية (تنعكس) خارج الوظيفة

استدعاء وظيفة جافاسكربت

عندما **function** سيتم تنفيذ التعليمات البرمجية الموجودة داخل جافاسكربت "يستدعيها" شيء ما.

استدعاء وظيفة

. لا يتم تنفيذ التعليمات البرمجية الموجودة داخل الوظيفة عند تعريف الوظيفة . يتم تنفيذ التعليمات البرمجية الموجودة داخل الوظيفة عند استدعاء الوظيفة . "من الشائع استخدام مصطلح " استدعاء دالة " بدلاً من " استدعاء دالة " . "ومن الشائع أيضاً قول " استدعاء وظيفة " أو " بدء وظيفة " أو " تنفيذ وظيفة " في هذا الكتاب، سوف نستخدم الاستدعاء ، لأنه يمكن استدعاء دالة جافاسكربت بدون استدعائها

استدعاء وظيفة كوظيفة

مثال

```
function Husini(a, b) {  
  return a * b;  
}  
Husini(10, 2); // Will return 20
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الوظيفة أعلاه لا تنتمي إلى أي كائن. ولكن في جافاسكربت يوجد دائماً كائن عام افتراضي.

نفسها، وبالتالي فإن HTML الكائن العام الافتراضي هو صفحة HTML، في HTML الوظيفة أعلاه "تنتمي" إلى صفحة

في المتصفح، كائن الصفحة هو نافذة المتصفح. الوظيفة المذكورة أعلاه تصبح تلقائياً وظيفة نافذة

ملحوظة

هذه طريقة شائعة لاستدعاء وظيفة جافاسكربت، ولكنها ليست ممارسة جيدة جداً.

يمكن للمتغيرات العامة أو الأساليب أو الوظائف إنشاء تعارضات في الأسماء وأخطاء في الكائن العام بسهولة

هما نفس الوظيفة window.Husini() و Husini():

مثال

```
function Husini(a, b) {  
  return a * b;  
}
```

```
window.Husini(10, 2); // Will also return 20
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

الكائن العام

الكائن العام **this** عندما يتم استدعاء دالة بدون كائن مالك، تصبح قيمة
في متصفح الويب، الكائن العام هو نافذة المتصفح
this: يُرجع هذا المثال كائن النافذة كقيمة

مثال

```
let x = Husini(); // x will be the window object

function Husini() {
  return this;
}
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

يؤدي استدعاء دالة كدالة عامة إلى جعل قيمة هذا كائنًا عالميًا
كمتغير إلى تعطل البرنامج بسهولة **window** قد يؤدي استخدام كائن

استدعاء وظيفة كطريقة

في جافاسكربت، يمكنك تعريف الوظائف كطرق كائن
له خاصيتين (الاسم)، **abib ect**) ينشئ المثال التالي كائنًا
:الأول واسم العائلة)، وطريقة (الاسم الكامل)

مثال

```
const abib ect = {
  firstName:"Habib",
  lastName: "Al Husini 😊😊😊",
  fullName: function () {
    return this.firstName + " " + this.lastName;
  }
}
abib ect.fullName(); // Will return "Abu Bakr Al-
Siddiq"
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

هو مالك **abib ect**. طريقة الاسم الكامل هي دالة. الوظيفة تنتمي إلى الكائن الوظيفة.

هو الكائن الذي "يمتلك" كود جافاسكربت. في هذه الحالة **this** الشيء المسمى **abib ect** هي **this** قيمة.

this: قم بتجريبه! قم بتغيير طريقة الاسم الكامل لإرجاع قيمة

مثال

```
const abib ect = {
  firstName:"Habib",
  lastName: "Al Husini 😊😊😊",
  fullName: function () {
    return this;
  }
}
```

```
// This will return [object Object] (the use object)
```

```
abib ect.fullName();
```

كتاب الكافي في جافاسكريبت الجزء الثالث أبو حبيب الحسيني

أن تكون قيمة الكائن نفسه **this** يؤدي استدعاء دالة كأسلوب كائن إلى

استدعاء وظيفة مع منشئ الوظيفة

المحجوزة، فهو استدعاء منشئ **new** إذا كان استدعاء دالة مسبقاً بالكلمة

يبدو أنك تقوم بإنشاء وظيفة جديدة، ولكن بما أن وظائف جافاسكريبت عبارة عن كائنات، فإنك تقوم بالفعل بإنشاء كائن جديد

مثال

```
// This is a function constructor:
```

```
function Husini(arg1, arg2) {  
  this.firstName = arg1;  
  this.lastName = arg2;  
}
```

```
// This creates a new object
```

```
const abib = new Husini("Habib", "Al Husini 😊😊😊");
```

```
// This will return "Habib"
```

```
abib .firstName;
```

كتاب الكافي في جافاسكربت الجزء الثالث ابو حبيب الحسيني

يؤدي استدعاء المنشئ إلى إنشاء كائن جديد. يرث الكائن الجديد الخصائص والأساليب من منشئه

.المحجوزة في المنشئ ليس لها قيمة **this** الكلمة

.الكائن الجديد الذي تم إنشاؤه عند استدعاء الوظيفة **this** ستكون قيمة

ابو حبيب الحسيني

ابو حبيب الحسينى

التكملة في الجزء الرابع

بإذن الله تعالى

أبو حبيب الحسيني