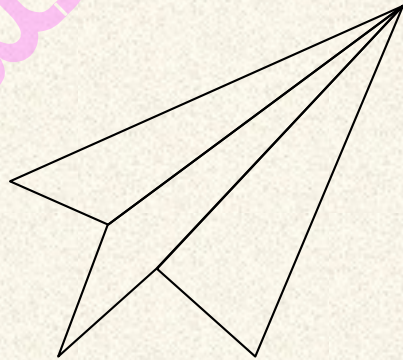


الكافى

فى

JavaScript

الجزء الثانى



أبو حبيب الحسينى

ابو ح



ملحوظة مهمة جداً

ان وجود الكلمات الانجليزية فى وسط الجمل العربية ينقل بعض الكلمات من مكانها فتظهر الجمل بشكل غير صحيح ويصعب فهما

مثال هذا الكلام

(بدلاً من إلقاء خطأ) `null` أو `undefined` كان الكائن `undefined` يُرجع عامل التشغيل إذا.?

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

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

ابو حبيب الحسينى

فهرس الكتاب

3	ملحوظة مهمة جدا
5	فهرس الكتاب
13	التحكم فى الوقت والتاريخ
14	ملحوظة
14	التاريخ وطرق الإخراج
14	طرق إنشاء كائنات التاريخ
17	الاعلان عن كائن التاريخ
17	طريقة اخرى لكائن التاريخ (نص التاريخ)
18	وظيفة اخراج (السنة، الشهر)
18	ملحوظة
19	استخدام 6، 4، 3، أو 2 من الأرقام
20	كيف الوصول الى القرن السابق
21	تقوم بتخزين التواريخ بالمللي ثانية
21	كيف الانشاء ب (ملي ثانية)
22	طرق ووظائف التاريخ
22	عرض التواريخ
23	استكمال مرجع التاريخ
24	تنسيقات التاريخ
24	طرق إدخال التاريخ
24	الإخراج الافتراضى للوقت والتاريخ
24	تواريخ ISO
25	(السنة والشهر) ISO
25	(السنة فقط) ISO
25	(التاريخ والوقت) ISO
26	المناطق الزمنية
26	كيف استخدام صيغ التواريخ المختصرة .
27	تحذيرات!
27	استخدام صيغ التواريخ الطويلة.
28	تحليل التواريخ
29	استكمال مرجع التاريخ
29	الحصول على طرق التاريخ
29	مُنشئ التاريخ
29	طرق التاريخ

ملاحظة.....	30
وظيفة getFullYear().....	30
تحذير!.....	31
وظيفة getMonth().....	31
ملحوظة 1.....	31
ملحوظة 2.....	32
وظيفة getDate().....	32
وظيفة getHours().....	33
وظيفة getMinutes().....	33
وظيفة getSeconds ().....	34
وظيفة getMilliseconds ().....	34
وظيفة getDay().....	35
ملحوظة 1.....	35
ملحوظة 2.....	35
وظيفة getTime().....	36
وظيفة Date.now().....	37
طرق UTC.....	37
وظيفة getTimezoneOffset ().....	38
استكمال مرجع التاريخ.....	38
ضبط طرق التاريخ.....	39
وظيفة setFullYear().....	39
وظيفة setMonth ().....	40
وظيفة setDate ().....	40
وظيفة setHours().....	40
وظيفة setMinutes().....	41
وظيفة setSeconds ().....	41
قارن التواريخ.....	41
كائن الرياضيات.....	42
الكلمة Math.....	42
خصائص الرياضيات (الثوابت).....	43
طرق الرياضيات.....	43
كيف تقرب الأرقام.....	43
التقريب التصاعدي.....	44
التقريب إلى التنازلي.....	44
كيف إزالة الكسور العشرية.....	44
طرق الاختبار.....	45
إيجاد القوة.....	45
إيجاد الجزر التربيعي.....	46
إيجاد القيمة المطلقة.....	46
طرق إيجاد زاوية الجيب.....	46
تابع طرق زاوية الجيب.....	46
Math.min() و Math.max().....	47

توليد الارقام العشوائية.....	47
Math.log() وظيفة و.....	48
Math.log2() وظيفة و.....	48
Math.log10() وظيفة و.....	49
طرق الرياضيات.....	49
كيف توليد رقم عشوائى.....	50
الأعداد الصحيحة العشوائية.....	51
تابع توليد الارقام العشوائية.....	52
الكلمات المنطقية.....	53
القيم المنطقية.....	53
الدالة بولين.....	53
المقارنات والشروط.....	53
كل شيء له "قيمة" يعتبر صحيح.....	54
كل شيء بدون "قيمة" يعتبر خطأ.....	54
Booleans تابع.....	56
المقارنات.....	57
عوامل المقارنة.....	58
كيف يمكن استخدام عوامل المقارنة.....	58
المعاملات المنطقية.....	58
العامل الشرطي (الثلاثي).....	59
مقارنة الأنواع المختلفة.....	59
كيف استخدام عامل الدمج الفارغ (؟؟).....	60
مشغل التسلسل الاختياري (؟).....	60
عبارات شرطية.....	61
كيف وضع شرط فى الكود.....	61
إذا الشرط صحيح افعل كذا والا افعل كذا.....	62
كلمة آخر إذا.....	63
صيغة اخرى لوضع الشروط.....	64
استخدام الكلمة سيوتش.....	64
الكلمة بريك.....	66
وضع الشرط الافتراضى.....	66
كتل ال اكواد المشتركة.....	68
تفاصيل التبديل.....	68
المقارنة الصارمة.....	68
للحلقة.....	69
حلقات.....	69
أنواع مختلفة من الحلقات.....	70
For الحلقة.....	70
البرامتر 1.....	71
البرامتر 2.....	71
البرامتر 3.....	72
نطاق الحلقة.....	72

حلقة بينما.....	73
حلقة For In.....	73
مثال اخر.....	74
التنفيذ على المصفوفة.....	74
forEach () على المصفوفة.....	75
تابع انواع الحلقات.....	76
دعم المتصفح.....	76
التكرار فوق مصفوفة.....	76
التكرار على نص.....	77
أثناء الحلقة.....	77
حلقة "بينما".....	77
حلقة "افعل أثناء كذا".....	78
مقارنة لحلقة بينما.....	79
كلمة التوقف من داخل الحلقة.....	80
كلمة الاستمرار.....	80
تسميات.....	80
عناصر التكرارية.....	81
للحلقة.....	82
التكرار.....	82
التكرار على نص.....	82
التكرار على مصفوفة.....	82
التكرار على مجموعة.....	83
التكرار على خريطة المصفوف.....	83
مجموعات.....	84
طرق المجموعة المحجوزة.....	84
كيفية إنشاء مجموعة.....	84
Set () وظيفه.....	85
وظيفة الإضافة.....	86
forEach() وظيفه.....	86
values() وظيفه.....	87
خرائط.....	87
كيفية إنشاء خريطة.....	87
new Map() وظيفه.....	88
set() وظيفه.....	88
وظيفة الحصول على قيمة.....	89
خاصية الحجم.....	89
وظيفة الحذف.....	89
الوظيفة has().....	89
كائنات مقارنة ب الخرائط.....	90
forEach() وظيفه.....	90
وظيفة الإدخالات.....	91
نوع المشغل.....	92

الكلمات البدائية.....	93
ايجاد النوع.....	93
تابع typeof.....	94
خاصية البناء.....	94
غير معرف.....	95
القيم الفارغة.....	96
خطا.....	96
الفرق بين غير محدد وفارغ.....	97
مثيل المشغل.....	97
المشغل الفارغ.....	98
تحويل نوع.....	98
تحويل النصوص إلى أرقام.....	98
طرق ووظائف الأرقام.....	99
المشغل الأحادي+.....	99
كيف تحويل الأرقام إلى نصوص.....	100
المزيد من الدوال.....	100
تحويل التواريخ إلى أرقام.....	100
تحويل التواريخ إلى نصوص.....	101
تحويل القيم المنطقية إلى أرقام.....	102
تحويل القيم المنطقية إلى نصوص.....	102
التحويل التلقائي للنوع.....	102
تحويل النص التلقائي.....	102
جدول تحويل الانواع فى.....	103
Bitwise مشغلي.....	104
أمثلة.....	104
يستخدم معاملات 32 بت.....	105
Bitwise AND.....	105
Bitwise OR.....	105
Bitwise XOR.....	106
Bitwise AND (&).....	106
Bitwise () أو.....	106
Bitwise XOR (^).....	107
Bitwise NOT (~).....	107
كيف إزاحة البت ليسار بمقدار (<<).....	107
علامة الازاحة الى اليمين (>).....	108
علامة التحول الأيمن (>>).....	108
كيف ترجمة الارقام الى لغة البينارى الثنائية.....	108
تحويل عشري إلى ثنائي.....	109
تحويل ثنائي إلى عشري.....	109
البرامترات العادية.....	110
ما هو البرامتر العادي؟.....	110
بناء الجملة.....	110

استخدام دوال النص.....	110
طرق البحث فى النصوص.....	111
احد طرق البحث الغير حساس لحالة الاحرف.....	111
كيف البحث و الاستبدال.....	112
طرق اخرى لاستبدال الغير حساس للحالة.....	112
هل لاحظت؟.....	112
معدّلات البرامتر العادي.....	113
خصائص البرامتر العادية.....	113
RegExp باستخدام كائن.....	114
اساليب الاختبار.....	114
تابع.....	114
أسبقية مشغل.....	115
قيم أسبقية المشغل.....	116
ارمي، وحاول أمسك وارسل النتيجة.....	118
سوف تحدث أخطاء!.....	118
عبيلة واديلة ☺.....	119
ماذا يحدث عند الاخطاء.....	119
التحكم فى الاخطاء.....	119
للتحقق من صحة الإدخال.....	120
التحقق من صحة HTML.....	121
البرامتر الاخير.....	121
كائن الخطأ.....	122
قيم اسم الخطأ.....	122
خطأ فى التقييم.....	122
خطأ فى النطاق.....	122
خطأ مرجعي.....	123
خطأ فى بناء الجملة.....	123
كيف معرفة نوع الخطا.....	124
كيف معرفة اخطا الروابط.....	124
خصائص كائن الخطأ غير القياسي.....	125
نطاق الكتلة.....	126
النطاق المحلي.....	126
نطاق الوظيفة.....	127
متغيرات العامة.....	127
النطاق العام.....	128
متغيرات.....	128
العامة التلقائيه.....	129
الوضع الصارم.....	129
المتغيرات العامة فى HTML.....	129
تحذير.....	130
عمر متغيرات.....	130
ما هى البرمترات فى الوظائف.....	130

مفهوم الرفع فى البرمجة.....	130
اعلان عن متغير.....	131
افضليات لـ const و Let.....	131
لا يتم رفع عمليات تهيئة فى.....	132
أعلن عن المتغيرات فى الأعلى!.....	133
استخدم الكلمة Strict.....	134
توجيه "الاستخدام الصارم".....	134
إعلان الوضع الصارم.....	134
استخدام صارم "؛ بناء الجملة".....	135
لماذا الوضع الصارم؟.....	136
الغير مسموح به فى الوضع الصارم.....	136
برهان المستقبل!.....	140
احترس!.....	140
هذه الكلمة المحجوزة.....	140
ما هذا؟.....	141
ملحوظة.....	141
الاشارة من داخل الدالة.....	141
تابع كلمة هذا.....	142
كلمة هذا فى الوضع العادى.....	142
كلمة هذا فى الوضع الصارم.....	142
كلمة هذا فى معالجات الأحداث.....	143
ربط وظيفة الكائن.....	143
ربط الوظيفة الصريحة.....	144
أنظر أيضا:.....	144
وظيفة الاقتراض.....	145
الأسبقية.....	145
الوظيفة المختصرة.....	146
this فى الوظائف المختصرة.....	147
دعم المتصفح.....	149
فئات.....	149
بناء الكلاسات.....	149
كيف استخدام الكلاس المنشأ.....	150
وظيفة البناء للكلاس او الكلاس.....	150
طرق الكلاس.....	151
دعم المتصفح.....	152
الوحدات.....	152
كيف تصدير الاجرائات.....	153
تسمية التصدير.....	153
التصدير الافتراضية.....	153
كيف الاستيراد.....	154
ملحوظة.....	155
ما هو جسون؟.....	155

جيسون (مثال)	155
لكائنات JSON يتم تقييم تنسيق	156
قواعد بناء جملة JSON	156
اسم وقيمة - JSON كلمات	156
JSON كائنات	156
مصفوفات جيسون	157
إلى كائن JSON تحويل نص	157
تصحيح الاكواد	158
مصحات	159
console.log() وظيفة	159
تحديد نقاط التوقف	160
الكلمة المحجوزة للمصحح	160
أدوات تصحيح أخطاء المتصفحات	160
كروم	160
فايرفوكس	161
حافة	161
الأوبرا	161
سفاري	161
هل كنت تعلم؟	161
مراجعات سريعة	162
الأسماء المتغيرة او المتغيرات	162
المساحات حول المشغلين	163
المسافة البادئة للكود	163
قواعد	163
قواعد الكائنات	164
طول الخط > 80	165
اتفاقيات التسمية	165
HTML تحميل في	166
HTML الوصول إلى عناصر	166
ملحقات الملفات	167
استخدم أسماء الملفات ذات الأحرف الصغيرة	167
أداء	167
تجنب المتغيرات العامة	168
أعلان المتغيرات المحلية	168
ميزة الاعلانات في الأعلى	168
تهيئة المتغيرات	169
قم بتعريف الكائنات باستخدام Const	169
قم بتعريف المصفوفات باستخدام const	170
اهم الكلاسات المستخدمة	170
احذر من تحويلات النوع التلقائية	171
استخدم === للمقارنة	171
استخدم إعدادات المعلمة الافتراضية	172

173	قم بإنهاء مفاتيحك باستخدام الإعدادات الافتراضية
173	لا تستخدم الأرقام والنصوص ككائنات
174	تجنب استخدام eval()
176	التكلمة فى الجزء الثالث
176	بإذن الله تعالى

أبو حبيب الحسينى

التحكم فى الوقت والتاريخ

تتيح لنا كائنات تاريخ جافاسكريبت التعامل مع التواريخ

أمثلة

```
const abibDate = new Date();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date("2022-03-25");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

ملحوظة

" كائنات التاريخ ثابتة. الساعة ليست قيد التشغيل وتستطيع ويتم تشغيلها بطرق كثيرة سنتطرق إليها ان شاء الله تعالى

التاريخ وطرق الإخراج

افتراضياً، ستستخدم جافاسكريبت المنطقة الزمنية للمتصفح يتم جلبها من الجهاز او الكمبيوتر او الموبايل وتعرض التاريخ كنصوص كاملة:

الأحد 19 نوفمبر 2023 الساعة 20:19:13 بتوقيت جرينتش+0200 (التوقيت الرسمي لأوروبا الشرقية)

. سوف تتعلم المزيد حول كيفية التحكم و العرض في جميع اجزاء التواريخ، والوقت باذن الله تعالى في هذا الكتاب

طرق إنشاء كائنات التاريخ

المُنشئ `new Date()` يتم إنشاء كائنات التاريخ باستخدام

هناك 9 طرق لإنشاء كائن تاريخ جديد:

new Date()

new Date(date string)

new Date(year,month)

new Date(year,month,day)

new Date(year,month,day,hours)

new Date(year,month,day,hours,minutes)

new Date(year,month,day,hours,minutes,seconds)

new Date(year,month,day,hours,minutes,seconds,ms)

new Date(milliseconds)

Name	وصف
constructor	تُرجع الدالة التي أنشأت النموذج الأولي لكائن التاريخ
getDate()	إرجاع يوم الشهر (من 1 إلى 31)
getDay()	إرجاع يوم الأسبوع (من 0 إلى 6)
getFullYear()	يرجع السنة
getHours()	إرجاع الساعة (من 0 إلى 23)
getMilliseconds()	إرجاع المللي ثانية (من 0-999)
getMinutes()	إرجاع الدقائق (من 0-59)
getMonth()	إرجاع الشهر (من 0 إلى 11)
getSeconds()	إرجاع الثواني (من 0 إلى 59)
getTime()	إرجاع عدد المللي ثانية منذ منتصف ليل 1 يناير 1970 وتاريخ محدد
getTimezoneOffset()	والتوقيت (UTC) إرجاع فارق التوقيت بين التوقيت العالمي المنسق المحلي، بالدقائق
getUTCDate()	إرجاع يوم الشهر حسب التوقيت العالمي (من 1 إلى 31)
getUTCDay()	إرجاع يوم الأسبوع حسب التوقيت العالمي (من 0 إلى 6)
getUTCFullYear()	إرجاع السنة حسب التوقيت العالمي
getUTCHours()	إرجاع الساعة حسب التوقيت العالمي (من 0 إلى 23)
getUTCMilliseconds()	إرجاع المللي ثانية، وفقاً للتوقيت العالمي (من 0 إلى 999)

<u>getUTCMinutes()</u>	إرجاع الدقائق حسب التوقيت العالمي (من 0 إلى 59)
<u>getUTCMonth()</u>	إرجاع الشهر حسب التوقيت العالمي (من 0 إلى 11)
<u>getUTCSeconds()</u>	إرجاع الثواني حسب التوقيت العالمي (من 0 إلى 59)
<u>getFullYear()</u>	بدلاً من ذلك <u>getFullYear()</u> إهمال. استخدم طريقة
<u>now()</u>	إرجاع عدد الملي ثانية منذ منتصف ليل 1 يناير 1970
<u>parse()</u>	يوزع نص تاريخ ويعيد عدد الملي ثانية منذ 1 يناير 1970
<u>prototype</u>	يسمح لك بإضافة خصائص ودوال إلى كائن
<u>setDate()</u>	يضبط اليوم من الشهر لكائن التاريخ
<u>setFullYear()</u>	يضبط سنة كائن التاريخ
<u>setHours()</u>	يضبط ساعة كائن التاريخ
<u>setMilliseconds()</u>	يضبط الملي ثانية لكائن التاريخ
<u>setMinutes()</u>	قم بتعيين دقائق كائن التاريخ
<u>setMonth()</u>	يضبط شهر كائن التاريخ
<u>setSeconds()</u>	يضبط الثواني لكائن التاريخ
<u>setTime()</u>	يضبط تاريخاً لعدد محدد من الملي ثانية بعد/قبل 1 يناير 1970
<u>setUTCDate()</u>	يضبط اليوم من الشهر لكائن التاريخ، وفقاً للتوقيت العالمي
<u>setUTCFullYear()</u>	يضبط سنة كائن التاريخ، وفقاً للتوقيت العالمي
<u>setUTCHours()</u>	يضبط ساعة كائن التاريخ، وفقاً للتوقيت العالمي
<u>setUTCMilliseconds()</u>	يضبط الملي ثانية لكائن التاريخ، وفقاً للتوقيت العالمي
<u>setUTCMinutes()</u>	قم بتعيين دقائق كائن التاريخ، وفقاً للتوقيت العالمي
<u>setUTCMonth()</u>	يضبط شهر كائن التاريخ، وفقاً للتوقيت العالمي
<u>setUTCSeconds()</u>	قم بتعيين الثواني لكائن التاريخ، وفقاً للتوقيت العالمي
<u>setYear()</u>	بدلاً من ذلك <u>setFullYear()</u> إهمال. استخدم طريقة
<u>toDateStr()</u>	يحول جزء التاريخ من كائن التاريخ إلى نص قابلة للقراءة
<u>toGMTStr()</u>	بدلاً من ذلك <u>toUTCStr()</u> إهمال. استخدم طريقة
<u>toISOStr()</u>	ISO إرجاع التاريخ كنص، باستخدام معيار
<u>toJSON()</u>	JSON إرجاع التاريخ كنص، بتنسيق تاريخ
<u>toLocaleDateStr()</u>	تقوم بإرجاع جزء التاريخ من كائن التاريخ كنص، باستخدام اصطلاحات اللغة
<u>toLocaleTimeStr()</u>	إرجاع الجزء الزمني من كائن التاريخ كنص، باستخدام اصطلاحات

	اللغة
<code>toLocaleString()</code>	يحول كائن التاريخ إلى نص، باستخدام اصطلاحات اللغة
<code>toString()</code>	تحويل كائن التاريخ إلى نص
<code>toTimeString()</code>	يحول الجزء الزمني من كائن التاريخ إلى نص
<code>toUTCString()</code>	يحول كائن التاريخ إلى نص، وفقاً للتوقيت العالمي
<code>UTC()</code>	إرجاع عدد المللي ثانية في تاريخ ما منذ منتصف ليل 1 يناير 1970 UTC، وفقاً لتوقيت
<code>valueOf()</code>	إرجاع القيمة الأولية لكائن التاريخ

الإعلان عن كائن التاريخ

: يقوم بإنشاء كائن تاريخ بالتاريخ والوقت الحاليين `new Date()`

(مثال ✓ ↓↓)

```
const abibDate = new Date();
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

طريقة أخرى لكائن التاريخ (نص التاريخ)

: ينشئ كائن تاريخ من نص تاريخ `new Date(date string)`

أمثلة

```
const abibDate = new Date("October 13, 2014 11:13:00");
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date("2022-03-25");
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

يتم وصف تنسيقات نص التاريخ في الفصل التالي

وظيفة اخراج (السنة، الشهر)

. يقوم بإنشاء كائن تاريخ بتاريخ ووقت محددين (`new Date(year, month, ...)`)

أرقام تحدد السنة والشهر واليوم والساعة والدقيقة والثانية والميلي ثانية (بهذا الترتيب) 7

(مثال ✓ ↓↓)

```
const abibDate = new Date(2018, 11, 24, 10, 33, 30, 0);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

ملحوظة

: جافاسكربت تحسب الأشهر من 0 إلى 11

. يناير = 0

. ديسمبر = 11

:لن يؤدي تحديد شهر أعلى من 11 إلى حدوث خطأ، ولكن سيتم إضافة الفائض إلى العام التالي

تحديد:

```
const abibDate = new Date(2018, 15, 24, 10, 33, 30);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

:بالضبط مثل

```
const abibDate = new Date(2019, 3, 24, 10, 33, 30);
```


كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

لن يؤدي تحديد يوم أعلى من الحد الأقصى إلى حدوث خطأ، ولكن سيتم إضافة الفائض إلى الشهر التالي

تحديد:

```
const abibDate = new Date(2018, 5, 35, 10, 33, 30);
```

بالتضبط مثل:

```
const abibDate = new Date(2018, 6, 5, 10, 33, 30);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

استخدام 6، 4، 3، أو 2 من الأرقام

أرقام تحدد السنة والشهر واليوم والساعة والدقيقة والثانية 6

(مثال ✓ ↓↓)

```
const abibDate = new Date(2018, 11, 24, 10, 33, 30);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

أرقام تحدد السنة والشهر واليوم والساعة والدقيقة 5

(مثال ✓ ↓↓)

```
const abibDate = new Date(2018, 11, 24, 10, 33);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

أرقام تحدد السنة والشهر واليوم والساعة 4

(مثال ✓ ↓↓)

```
const abibDate = new Date(2018, 11, 24, 10);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

أرقام تحدد السنة والشهر واليوم 3

(مثال ✓ ↓↓)

```
const abibDate = new Date(2018, 11, 24);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

رقمان يحددان السنة والشهر

(مثال ✓ ↓↓)

```
const abibDate = new Date(2018, 11);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

لا يمكنك حذف الشهر. إذا قمت بتوفير معلمة واحدة فقط، فسيتم التعامل معها على أنها ميلي ثانية.

(مثال ✓ ↓↓)

```
const abibDate = new Date(2018);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

كيف الوصول الى القرن السابق

XX: سيتم تفسير السنوات المكونة من رقم واحد أو رقمين على أنها 19

(مثال ✓ ↓↓)

```
const abibDate = new Date(99, 11, 24);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

(مثال ↓↓)

```
const abibDate = new Date(9, 11, 24);
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

تقوم بتخزين التواريخ بالملي ثانية

تقوم جافاسكربت بتخزين التواريخ بعدد الملي ثانية منذ 01 يناير 1970

. الوقت صفر هو 01 يناير 1970 00:00:00 بالتوقيت العام

يوم واحد (24 ساعة) هو 86400000 ملي ثانية

الوقت الآن هو: 1700417953322 ملي ثانية بعد 01 يناير 1970

كيف الإنشاء ب (ملي ثانية)

ينشئ كائن تاريخ جديدًا بالملي ثانية بالإضافة إلى وقت الصفر (new Date(milliseconds))

أمثلة

:يناير 1970 بالإضافة إلى 100000000000 ملي ثانية هو 01

```
const abibDate = new Date(100000000000);
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

:يناير 1970 ناقص 100000000000 ملي ثانية هو 01

```
const abibDate = new Date(-10000000000);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

يناير 1970 زائد 24 ساعة هو 01

```
const abibDate = new Date(24 * 60 * 60 * 1000);
```

```
// or
```

```
const abibDate = new Date(86400000);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

يناير 1970 زائد 0 مللي ثانية هو 01

```
const abibDate = new Date(0);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

طرق ووظائف التاريخ

عندما يتم إنشاء كائن التاريخ، هناك عدد من الطرق التي تسمح لك بالعمل عليه.

تسمح لك دوال التاريخ بالحصول على كائنات التاريخ للسنة والشهر واليوم والساعة والدقيقة والثانية والمللي ثانية وتعيينها، (العام أو بتوقيت جرينتش) (UTC) باستخدام التوقيت المحلي أو التوقيت العام.

عرض التواريخ

ستقوم جافاسكريبت (افتراضياً) بإخراج التواريخ بتنسيق نصوص كاملة

(مثال ↓↓↓)

```
19 2023 20:19:13 GMT+0200 (Eastern European Standard Time)
```


كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

عند عرض كائن تاريخ في ، يتم تحويله تلقائيًا إلى نص باستخدام الوظيفة **toString()**.

(مثال ✓ ↓↓)

```
const abibDate = new Date();
abibDate.toString();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

بتحويل التاريخ إلى تنسيق أكثر قابلية للقراءة **toDateString()** تقوم الوظيفة

(مثال ✓ ↓↓)

```
const abibDate = new Date();
abibDate.toDateString();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

بتحويل التاريخ إلى نص باستخدام معيار **toUTCString()** تقوم الوظيفة

(مثال ✓ ↓↓)

```
const abibDate = new Date();
abibDate.toUTCString();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

بتحويل التاريخ إلى نص باستخدام معيار **toISOString()** تقوم الوظيفة

(مثال ✓ ↓↓)

```
const abibDate = new Date();
abibDate.toISOString();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

استكمال مرجع التاريخ

للحصول على مرجع كامل للتاريخ، انتقل إلى

المرجع الكامل لجافاسكربت

يحتوي المرجع على أوصاف وأمثلة لجميع خصائص ودوال كل شى فى جافاسكربت

تنسيقات التاريخ

طرق إدخال التاريخ

هناك بشكل عام ثلاثة أنواع من تنسيقات إدخال التاريخ في جافاسكربت

يكتب	مثال
تاريخ الايزو	"2015-03-25" (The International Standard)
التاريخ القصير	"03/25/2015"
تاريخ طويل	"Mar 25 2015" or "25 Mar 2015"

معييرًا صرًا في جافاسكربت ISO يتبع تنسيق

التنسيقات الأخرى ليست محددة بشكل جيد وقد تكون خاصة بالمتصفح

الإخراج الافتراضي للوقت والتاريخ

بغض النظر عن تنسيق الإدخال، ستقوم جافاسكربت (افتراضيًا) بإخراج التواريخ بتنسيق نصوص كاملة مثل هذا الإخراج

Sun Nov 19 2023 20:19:41 GMT+0200 (Eastern European Standard Time)

تواريخ ISO

هو المعيار الدولي لتمثيل التواريخ والأوقات ISO 8601

أيضًا تنسيق التاريخ في جافاسكربت المفضل (YYYY-MM-DD) ISO 8601 يعد بناء جملة

(التاريخ الكامل) (↓↓↓ مثال ✓)

```
const abibDate = new Date("2015-03-25");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

سيكون التاريخ المحسوب مرتبطاً بمنطقتك الزمنية اعتماداً على منطقتك الزمنية، في الكمبيوتر او الجهاز اللوحى او الموبايل ستختلف النتيجة أعلاه بين 24 و25 مارس طبقاً لنوع المتصفح .

(السنة والشهر) ISO

(YYYY-MM) دون تحديد اليوم ISO يمكن كتابة تواريخ

(↓↓↓ مثال ✓)

```
const abibDate = new Date("2015-03");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

ستختلف المناطق الزمنية النتيجة المذكورة أعلاه بين 28 فبراير و01 مارس

(السنة فقط) ISO

(YYYY) بدون الشهر واليوم ISO يمكن كتابة تواريخ

(↓↓↓ مثال ✓)

```
const abibDate = new Date("2015");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

ستختلف المناطق الزمنية النتيجة المذكورة أعلاه بين 31 ديسمبر 2014 و01 يناير 2015

ISO (التاريخ والوقت)

(YYYY-MM-DDTHH:MM:SSZ) مع إضافة الساعات والدقائق والثواني ISO يمكن كتابة تواريخ

(مثال ✓)

```
const abibDate = new Date("2015-03-25T12:00:00Z");
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

كبير T يتم فصل التاريخ والوقت بحرف

Z. بالحرف الكبير (UTC) يتم تعريف التوقيت العام المنسق

بدلاً من ذلك -HH:MM أو +HH:MM وأضف Z قم بإزالة UTC، إذا كنت تريد تعديل الوقت بالنسبة إلى

(مثال ✓)

```
const abibDate = new Date("2015-03-25T12:00:00-06:30");
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

هو نفس توقيت جرينتش (توقيت جرينتش) (التوقيت العام المنسق) UTC

المناطق الزمنية

عند تحديد تاريخ، دون تحديد المنطقة الزمنية، ستستخدم جافاسكربت المنطقة الزمنية للمتصفح أو الكمبيوتر

عند الحصول على تاريخ، دون تحديد المنطقة الزمنية، يتم تحويل النتيجة إلى المنطقة الزمنية للمتصفح أو الكمبيوتر

التوقيت) CDT فسيتم تحويل التاريخ/الوقت إلى GMT (توقيت جرينتش) بمعنى آخر: إذا تم إنشاء التاريخ/الوقت بتوقيت
إذا كان المستخدم يتصفح من وسط الولايات المتحدة (الصيفي لوسط الولايات المتحدة)

. كيف استخدام صيغ التواريخ المختصرة

كما يلي "MM/DD/YYYY" تتم كتابة التواريخ القصيرة باستخدام الصيغة

(مثال ↓↓)

```
const abibDate = new Date("03/25/2015");
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

تحذيرات

:في بعض المتصفحات، قد تؤدي الأشهر أو الأيام التي لا تحتوي على أصفار بادئة إلى حدوث خطأ

```
const abibDate = new Date("2015-3-25");
```

غير محدد "YYYY/MM/DD" سلوك

NaN. ستحاول بعض المتصفحات القديمة تخمين التنسيق. سيعود بالتاريخ ناقص

```
const abibDate = new Date("2015/03/25");
```

غير محدد أيضًا "DD-MM-YYYY" سلوك

NaN. ستحاول بعض المتصفحات القديمة تخمين التنسيق. سيعود البعض

```
const abibDate = new Date("25-03-2015");
```

استخدام صيغ التواريخ الطويلة

:كما يلي "MMM DD YYYY" غالبًا ما تتم كتابة التواريخ الطويلة باستخدام هذه الصيغة

(مثال ↓↓)

```
const abibDate = new Date("Mar 25 2015");
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

يمكن أن يكون الشهر واليوم بأي ترتيب

(مثال ↓↓) ✓

```
const abibDate = new Date("25 Mar 2015");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

ويمكن كتابة الشهر كاملاً (يناير)، أو مختصراً

(مثال ↓↓) ✓

```
const abibDate = new Date("January 25 2015");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

(مثال ↓↓) ✓

```
const abibDate = new Date("Jan 25 2015");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

يتم تجاهل الفواصل. الأسماء غير حساسة لحالة الأحرف

(مثال ↓↓) ✓

```
const abibDate = new Date("JANUARY, 25, 2015");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

تحليل التواريخ

لتحويلها إلى ميلي ثانية (`Date.parse()`) إذا كان لديك نص تاريخ صالح، يمكنك استخدام الوظيفة

إرجاع عدد الملي ثانية بين التاريخ و1 يناير 1970 (`Date.parse()`)

(مثال ↓↓) ✓

```
let msec = Date.parse("March 21, 2012");
```


يمكنك بعد ذلك استخدام عدد المللي ثانية لتحويله إلى كائن تاريخ

(مثال ✓ ↓↓)

```
let msec = Date.parse("March 21, 2012");  
const abibDate = new Date(msec);
```

استكمال مرجع التاريخ

للحصول على مرجع كامل للتاريخ، انتقل إلى

المرجع الكامل لجافاسكريبت

الحصول على طرق التاريخ

مُنشئ التاريخ

`new Date()` في جافاسكريبت، يتم إنشاء كائنات التاريخ باستخدام

إرجاع كائن تاريخ بالتاريخ والوقت الحاليين `new Date()`.

الحصول على الوقت الحالي

```
const date = new Date();
```

طرق التاريخ

Method	وصف
<code>getFullYear()</code>	احصل على السنة كرقم مكون من أربعة أرقام (yyyy)
<code>getMonth()</code>	الحصول على الشهر كرقم (0-11)
<code>getDate()</code>	الحصول على اليوم كرقم (1-31)
<code>getDay()</code>	احصل على أيام الأسبوع كرقم (0-6)
<code>getHours()</code>	الحصول على ساعة (0-23)
<code>getMinutes()</code>	احصل على دقيقة (0-59)
<code>getSeconds()</code>	احصل على الثانية (0-59)
<code>getMilliseconds()</code>	الحصول على ميلي ثانية (0-999)
<code>getTime()</code>	الحصول على الوقت (ملي ثانية منذ 1 يناير 1970)

تُرجع طرق الحصول المذكورة أعلاه التوقيت المحلي.

في أسفل هذه الصفحة (UTC) تم توثيق التوقيت العام.

ملاحظة

يلجأ المعلومات من كائنات التاريخ الموجودة `get` تقوم دوال.

"في كائن التاريخ، يكون الوقت ثابتًا. "الساعة" ليست "فيد التشغيل

الوقت في كائن التاريخ ليس هو نفس الوقت الحالي.

`getFullYear()` وظيفة

يلجأ سنة التاريخ كرقم مكون من أربعة أرقام `getFullYear()` تقوم الوظيفة

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getFullYear();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();  
abibDate.getFullYear();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

! تحذير

`getFullYear()` قد يستخدم كود جافاسكريبت القديم الوظيفة غير القياسية

.بإرجاع سنة مكونة من رقمين `getFullYear()` من المفترض أن تقوم

لا تستخدمها. `getFullYear()` تم إهمالها

`getMonth()` وظيفة

.بإرجاع شهر التاريخ كرقم (0-11) `getMonth()` تقوم الوظيفة

ملحوظة 1

...، نعيد للأهمية في جافاسكريبت، يناير هو الشهر رقم 0، فبراير هو الشهر رقم 1

وأخيراً، ديسمبر هو الشهر رقم 11

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getMonth();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();  
abibDate.getMonth();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

ملحوظة 2

يمكنك استخدام مجموعة من الأسماء لإرجاع الشهر كاسم:

أمثلة

```
const months =  
["January", "February", "March", "April", "May", "June", "July", "August", "September",  
"October", "November", "December"];
```

```
const abibDate = new Date("2021-03-25");  
let month = months[abibDate.getMonth()];
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const months =  
["January", "February", "March", "April", "May", "June", "July", "August", "September",  
"October", "November", "December"];
```

```
const abibDate = new Date();  
let month = months[abibDate.getMonth()];
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

وظيفة getDate()

إرجاع يوم التاريخ كرقم (1-31) getDate() تقوم الوظيفة

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getDate();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();  
abibDate.getDate();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

وظيفة getHours()

إرجاع ساعات التاريخ كرقم (0-23) getHours() تقوم الوظيفة

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getHours();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();  
abibDate.getHours();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

وظيفة getMinutes()

يُرجع دقائق التاريخ كرقم (0-59) `getMinutes()` تقوم الوظيفة

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getMinutes();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();  
abibDate.getMinutes();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

وظيفة getSeconds ().

يُرجع ثواني التاريخ كرقم (0-59) `getSeconds()` تقوم الوظيفة

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getSeconds();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();  
abibDate.getSeconds();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

().getMilliseconds وظيفة

:إرجاع المللي ثانية من التاريخ كرقم (0-999) `getMilliseconds()` تقوم الوظيفة

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getMilliseconds();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();  
abibDate.getMilliseconds();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

().getDay وظيفة

:إرجاع يوم الأسبوع من التاريخ كرقم (0-6) `getDay()` تقوم الوظيفة

ملحوظة 1

.في جافاسكريبت، اليوم الأول من الأسبوع (اليوم 0) هو الأحد

. بعض دول العام تعتبر أول يوم في الأسبوع هو يوم الاثنين والبعض يعتبره السبت تستطيع تغييره

أمثلة

```
const abibDate = new Date("2021-03-25");  
abibDate.getDay();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const abibDate = new Date();
abibDate.getDay();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

ملحوظة 2

تُرجع أيام الأسبوع كاسم `getDay()`، يمكنك استخدام مجموعة من الأسماء

أمثلة

```
const days =
["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

const abibDate = new Date("2021-03-25");
let day = days[abibDate.getDay()];
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

```
const days =
["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

const abibDate = new Date();
let day = days[abibDate.getDay()];
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

getTime() وظيفة

عدد المللي ثانية منذ 1 يناير 1970 `getTime()` تُرجع الوظيفة

أمثلة

```
const abibDate = new Date("1970-01-01");
abibDate.getTime();
```



```
const abibDate = new Date("2021-03-25");  
abibDate.getTime();
```

```
const abibDate = new Date();  
abibDate.getTime();
```

Date.now() وظيفة

تُرجع عدد المللي ثانية منذ 1 يناير 1970 `Date.now()`.

أمثلة

```
let ms = Date.now();
```

حسب عدد السنوات منذ 01/01/1970:

```
const minute = 1000 * 60;  
const hour = minute * 60;  
const day = hour * 24;  
const year = day * 365;
```

```
let years = Math.round(Date.now() / year);
```

هي وظيفة ثابتة لكائن التاريخ `Date.now()`.

لا يمكنك استخدامه في كائن تاريخ مثل `myDate.now()`.

`Date.now()` بناء الجملة هو دائما

طرق UTC

Method	Same As	وصف
<code>getUTCDate()</code>	<code>getDate()</code>	UTC إرجاع تاريخ
<code>getUTCFullYear()</code>	<code>getFullYear()</code>	UTC إرجاع سنة
<code>getUTCMonth()</code>	<code>getMonth()</code>	UTC إرجاع شهر
<code>getUTCDay()</code>	<code>getDay()</code>	UTC إرجاع يوم
<code>getUTCHours()</code>	<code>getHours()</code>	UTC إرجاع ساعة
<code>getUTCMinutes()</code>	<code>getMinutes()</code>	UTC إرجاع دقائق
<code>getUTCSeconds()</code>	<code>getSeconds()</code>	UTC إرجاع ثواني
<code>getUTCMilliseconds()</code>	<code>getMilliseconds()</code>	UTC إرجاع ميلي ثانية

(التوقيت العام المنسق) UTC تستخدم طرق الوقت العادية

(توقيت غرينتش) GMT هو نفس توقيت UTC توقيت

إلى 24 ساعة (UTC) يمكن أن يصل الفرق بين التوقيت المحلي والتوقيت العام المنسق

الوقت المحلي؟

التوقيت العام المنسق؟

`getTimezoneOffset()` وظيفة

UTC: إرجاع الفرق (بالدقائق) بين التوقيت المحلي وتوقيت `getTimezoneOffset()` تقوم الوظيفة

(مثال) ↓↓↓

```
let diff = abibDate.getTimezoneOffset();
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

تعيين طرق التاريخ

إمكانية تعيين قيم التاريخ (السنوات، الأشهر، الأيام، الساعات، الدقائق، الثواني، المللي Set Date تتيح لك دوال ثانية) لكائن التاريخ.

ضبط طرق التاريخ

يتم استخدام دوال تعيين التاريخ لتعيين جزء من التاريخ:

Method	وصف
setDate()	تحديد اليوم كرقم (1-31)
setFullYear()	اضبط السنة (الشهر واليوم اختياريًا)
setHours()	ضبط الساعة (0-23)
setMilliseconds()	ضبط المللي ثانية (0-999)
setMinutes()	ضبط الدقائق (0-59)
setMonth()	ضبط الشهر (0-11)
setSeconds()	ضبط الثواني (0-59)
setTime()	ضبط الوقت (ملي ثانية منذ 1 يناير 1970)

setFullYear() وظيفة

سنة كائن التاريخ. في هذا ال (✓ مثال ↓↓) حتى عام 2020 setFullYear() تحدد الوظيفة

(✓ مثال ↓↓)

```
const abibDate = new Date();
abibDate.setFullYear(2020);
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

: يمكن للوظيفة تحديد الشهر واليوم اختياريًا setFullYear()

(مثال ↓↓)

```
const abibDate = new Date();  
abibDate.setFullYear(2020, 11, 3);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

setMonth () وظيفة

شهر كائن التاريخ (11-0) setMonth() تحدد الوظيفة

(مثال ↓↓)

```
const abibDate = new Date();  
abibDate.setMonth(11);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

setDate () وظيفة

يوم كائن التاريخ (31-1) setDate() تحدد الوظيفة

(مثال ↓↓)

```
const abibDate = new Date();  
abibDate.setDate(15);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

يمكن أيضًا استخدام الوظيفة لإضافة أيام إلى تاريخ setDate()

(مثال ↓↓)

```
const abibDate = new Date();  
abibDate.setDate(abibDate.getDate() + 50);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

إذا أردت إضافة الأيام إلى إراحة الشهر أو السنة، فسيتم التعامل مع التغييرات تلقائيًا بواسطة كائن التاريخ

setHours() وظيفة

ساعات كائن التاريخ (23-0) **setHours()** تحدد الوظيفة

(مثال ✓ ↓↓)

```
const abibDate = new Date();  
abibDate.setHours(22);
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

setMinutes() وظيفة

بتعيين دقائق كائن التاريخ (59-0) **setMinutes()** تقوم الوظيفة

(مثال ✓ ↓↓)

```
const abibDate = new Date();  
abibDate.setMinutes(30);
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

setSeconds () . وظيفة

بتعيين الثواني لكائن التاريخ (59-0) **setSeconds()** تقوم الوظيفة

(مثال ✓ ↓↓)

```
const abibDate = new Date();  
abibDate.setSeconds(30);
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

قارن التواريخ

يمكن بسهولة مقارنة التواريخ

يقارن ال (✓ مثال ↓↓) التالي تاريخ اليوم بتاريخ 14 يناير 2100

(✓ مثال ↓↓)

```
let abib = "";  
const today = new Date();  
const someday = new Date();  
someday.setFullYear(2100, 0, 14);
```

```
if (someday > today) {  
  abib = "Today is before January 14, 2100."  
} else {  
  abib = "Today is after January 14, 2100."  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

نعيد للتذكير جافاسكربت تحسب الأشهر من 0 إلى 11. يناير هو 0. ديسمبر هو 11

كائن الرياضيات

إجراء مهام رياضية على الأرقام Math يتيح لك كائن

(✓ مثال ↓↓)

```
Math.PI;
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

الكلمة Math

على مُنشئ **Math** على عكس الكائنات الأخرى، لا يحتوي كائن

ثابت **Math** كائن

يمكن استخدام كافة الدوال والخصائص دون إنشاء كائن رياضي أولاً

خصائص الرياضيات (الثوابت)

Math.property: بناء جملة أي خاصية رياضية هو

توفر جافاسكريبت 8 ثوابت رياضية يمكن الوصول إليها كخصائص رياضية

(مثال ↴)

```
Math.E // returns Euler's number
Math.PI // returns PI
Math.SQRT2 // returns the square root of 2
Math.SQRT1_2 // returns the square root of 1/2
Math.LN2 // returns the natural logarithm of 2
Math.LN10 // returns the natural logarithm of 10
Math.LOG2E // returns base 2 logarithm of E
Math.LOG10E // returns base 10 logarithm of E
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

طرق الرياضيات

Math.method(number): بناء جملة الرياضيات بأي وظيفة هو

هناك أربع طرق شائعة لتقريب رقم إلى عدد صحيح

كيف تقريب الأرقام

إرجاع أقرب عدد صحيح **Math.round(x)**

أمثلة

Math.round(4.6);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Math.round(4.5);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Math.round(4.4);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

التقريب التصاعدي

تقريب إلى أقرب عدد صحيح x تُرجع قيمة **Math.ceil(x)**

(مثال ↓↓↓)

Math.ceil(4.9);

Math.ceil(4.7);

Math.ceil(4.4);

Math.ceil(4.2);

Math.ceil(-4.2);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

التقريب إلى التنازلي

تقريب إلى أقرب عدد صحيح لها x تُرجع قيمة **Math.floor(x)**

(مثال ↓↓↓)

Math.floor(4.9);

Math.floor(4.7);

Math.floor(4.4);

Math.floor(4.2);

Math.floor(-4.2);

كيف ازالة الكسور العشرية

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

(مثال $\Downarrow\Downarrow$)

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

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

طرق الاختبار

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

(مثال $\Downarrow\Downarrow$)

```
Math.sign(-4);  
Math.sign(0);  
Math.sign(4);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

ايجاد القوة

Math.pow(x, y): إرجاع قيمة x إلى قوة y

(مثال ↓↓)

Math.pow(8, 2);

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

ايجاد الجذر التربيعي

Math.sqrt(x): إرجاع الجذر التربيعي لـ x :

(مثال ↓↓)

Math.sqrt(64);

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

ايجاد القيمة المطلقة

Math.abs(x): تُرجع القيمة المطلقة (الإيجابية) لـ x :

(مثال ↓↓)

Math.abs(-4.7);

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

طرق ايجاد زاوية الجيب

Math.sin(x): (المعطى بالراديان) x إرجاع جيب الزاوية (قيمة بين -1 و 1) للزاوية x .

إذا كنت تريد استخدام الدرجات بدلاً من الراديان، فيجب عليك تحويل الدرجات إلى راديان:

$x \text{ PI} / 180$ = الزاوية بالراديان = الزاوية بالدرجات

(مثال ↓↓)

Math.sin(90 * Math.PI / 180); / returns 1 (the sine of 90 degrees)

تابع طرق زاوية الجيب

(المعطى بالراديان) x إرجاع جيب التمام (قيمة بين -1 و 1) للزاوية $\text{Math.cos}(x)$.
إذا كنت تريد استخدام الدرجات بدلاً من الراديان، فيجب عليك تحويل الدرجات إلى راديان:
 $x \text{ PI} / 180$ = الزاوية بالراديان = الزاوية بالدرجات

(مثال ↓↓) ✓

```
Math.cos(0 * Math.PI / 180); // returns 1 (the cos of 0 degrees)
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Math.min() و Math.max()

استخدامها للعثور على أدنى أو أعلى قيمة في قائمة الوسائط $\text{Math.max}()$ ويمكن $\text{Math.min}()$:

(مثال ↓↓) ✓

```
Math.min(0, 150, 30, 20, -8, -200);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

(مثال ↓↓) ✓

```
Math.max(0, 150, 30, 20, -8, -200);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

توليد الأرقام العشوائية

إرجاع رقم عشوائي بين 0 (شاملاً) و 1 (حصرياً) $\text{Math.random}()$:

(مثال ↓↓)

Math.random();

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

في الفصل التالي من هذا الكتاب **Math.random()** سوف تتعلم المزيد عنها

Math.log() وظيفة

Math.log(x) X. إرجاع اللوغاريتم الطبيعي لـ

تُرجع اللوغاريتم الطبيعي الوقت اللازم للوصول إلى مستوى معين من الزيادة

أمثلة

Math.log(1);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Math.log(2);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Math.log(3);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

توأمان **Math.E** و **Math.log()**

لنحصل على **Math.E** ؟ 10 كم مرة يجب أن نضرب

Math.log(10);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Math.log2() وظيفة

$\text{Math.log2}(x)$ إرجاع اللوغاريتم الأساسي 2 لـ x .

كم مرة يجب أن نضرب 2 لنحصل على 8؟

Math.log2(8);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Math.log10() وظيفة

$\text{Math.log10}(x)$ إرجاع اللوغاريتم الأساسي 10 لـ x .

كم مرة يجب أن نضرب 10 لنحصل على 1000؟

Math.log10(1000);

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

طرق الرياضيات

Method	وصف
abs(x)	x إرجاع القيمة المطلقة لـ
acos(x)	بوحدة الراديان، x إرجاع قوس جيب التمام لـ
acosh(x)	x إرجاع قوس جيب التمام الزائدي لـ
asin(x)	بالراديان x إرجاع قوس جيب
asinh(x)	x إرجاع قوس الجيب الزائدي لـ
atan(x)	راديان $\text{PI}/2$ و $-\text{PI}/2$ كقيمة رقمية بين x إرجاع ظل الزاوية لـ
atan2(y, x)	إرجاع قوس الظل لحاصل وسيطاته
atanh(x)	x إرجاع قوس الظل الزائدي لـ

cbrt(x)	X إرجاع الجذر التكعيبي لـ
ceil(x)	مع تقريبه لأعلى إلى أقرب عدد صحيح، X إرجاع
cos(x)	(بالراديان X) إرجاع جيب تمام
cosh(x)	X إرجاع جيب تمام الزائدي لـ
exp(x)	Ex ترجع قيمة
floor(x)	مع تقريبه للأسفل إلى أقرب عدد صحيح، X إرجاع
log(x)	X لـ (E الأساس) إرجاع اللوغاريتم الطبيعي
max(x, y, z, ..., n)	إرجاع الرقم ذو القيمة الأعلى
min(x, y, z, ..., n)	إرجاع الرقم ذو القيمة الأقل
pow(x, y)	إلى قوة X إرجاع قيمة y
random()	إرجاع رقم عشوائي بين 0 و 1
round(x)	إلى أقرب عدد صحيح X تقريب
sign(x)	سالبة أو فارغة أو موجبة (-1، 0، 1) X تُرجع إذا كانت
sin(x)	(بالراديان X) إرجاع جيب الزاوية لـ
sinh(x)	X إرجاع جيب الزاوية الزائدي لـ
sqrt(x)	X إرجاع الجذر التربيعي لـ
tan(x)	إرجاع ظل الزاوية
tanh(x)	إرجاع الظل الزائدي لرقم
trunc(x)	(X) إرجاع الجزء الصحيح من الرقم

الارقام العشوائية

كيف توليد رقم عشوائي

إرجاع رقم عشوائي بين 0 (شاملاً) و 1 (حصرياً) **Math.random()**:

(مثال ↓↓) ✓

```
// Returns a random number:  
Math.random();
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

تقوم دائمًا بإرجاع رقم أقل من 1 `Math.random()`.

الأعداد الصحيحة العشوائية

يمكن استخدامها لإرجاع أعداد صحيحة عشوائية `Math.floor()` المستخدمة مع `Math.random()`.

لا يوجد شيء اسمه أعداد صحيحة وأعداد خاطئة في جافاسكربت.

نحن نتحدث هنا عن أرقام بدون كسور عشرية.

(مثال ↓↓) ✓

```
// Returns a random integer from 0 to 9:  
Math.floor(Math.random() * 10);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

(مثال ↓↓) ✓

```
// Returns a random integer from 0 to 10:  
Math.floor(Math.random() * 11);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

(مثال ↓↓) ✓

```
// Returns a random integer from 0 to 99:  
Math.floor(Math.random() * 100);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

(مثال ↓↓) ✓

```
// Returns a random integer from 0 to 100:  
Math.floor(Math.random() * 101);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

(مثال ↓↓) ✓

```
// Returns a random integer from 1 to 10:  
Math.floor(Math.random() * 10) + 1;
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

(مثال ↓↓) ✓

```
// Returns a random integer from 1 to 100:  
Math.floor(Math.random() * 100) + 1;
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

تابع توليد الأرقام العشوائية

كما ترون من الأمثلة أعلاه، قد يكون من الجيد إنشاء دالة عشوائية مناسبة لاستخدامها في جميع أغراض الأعداد الصحيحة العشوائية.

تقوم دالة جافاسكريبت هذه دائمًا بإرجاع رقم عشوائي بين الحد الأدنى (المضمن) والحد الأقصى (المستبعد):

(مثال ↓↓) ✓

```
function abibgetRndInteger(min, max) {  
  return Math.floor(Math.random() * (max - min) ) + min;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

تقوم دالة جافاسكريبت هذه دائمًا بإرجاع رقم عشوائي بين الحد الأدنى والحد الأقصى (كلاهما متضمن):

(مثال ↓↓) ✓

```
function abibgetRndInteger(min, max) {  
  return Math.floor(Math.random() * (max - min + 1)) + min;  
}
```

الكلمات المنطقية

. false أو true: إحدى القيمتين Boolean يمثل جافاسكربت

القيم المنطقية

في كثير من الأحيان، في البرمجة، ستحتاج إلى نوع كلمات يمكن أن يحتوي على قيمة واحدة فقط من قيمتين، مثل

- نعم / لا
- تشغيل / إيقاف
- خطأ صحيح

. ولهذا السبب، تحتوي جافاسكربت على نوع كلمات منطقي . يمكن أن تأخذ القيم صحيحة أو خاطئة فقط

الدالة بوليين

:الدالة لمعرفة ما إذا كان البرامتر (أو المتغير) صحيحًا Boolean() يمكنك استخدام

(مثال ↓↓) ✓

```
Boolean(10 > 9)
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

:أو حتى أسهل

(مثال ↓↓) ✓

```
(10 > 9)
```

```
10 > 9
```

المقارنات والشروط

نظرة عامة كاملة على عوامل المقارنة JS Comparisons يقدم الفصل

نظرة عامة كاملة على الكلمات الشرطية JS يقدم الفصل شروط

نوهنا بعض الأمثلة

Operator	وصف	مثال
==	يساوي	if (day == "Monday")
>	أكثر من	if (salary > 9000)
<	أقل من	if (age < 18)

القيمة المنطقية للتعبير هي الأساس لجميع مقارنات وشروط جافاسكربت

كل شيء له "قيمة" يعتبر صحيح

أمثلة

- 100
- 3.14
- 15
- "Easy-to"
- "false"
- 7 + 1 + 3.14

كل شيء بدون "قيمة" يعتبر خطأ

: القيمة المنطقية 0 (صفر) خاطئة

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

: القيمة المنطقية لـ 0 (ناقص الصفر) خاطئة

```
let x = -0;  
Boolean(x);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

: القيمة المنطقية لـ "" (نص فارغة) غير صحيحة

```
let x = "";  
Boolean(x);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

: القيمة المنطقية غير المحددة غير صحيحة

```
let x;  
Boolean(x);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

: القيمة المنطقية للقيمة الخالية خاطئة

```
let x = null;  
Boolean(x);
```

false : القيمة المنطقية للخطأ هي

```
let x = false;  
Boolean(x);
```

: خاطئة NaN القيمة المنطقية لـ

```
let x = 10 / "Hallo";  
Boolean(x);
```

تابع Booleans

عادةً ما تكون القيم المنطقية لجافاسكريبت عبارة عن قيم بدائية تم إنشاؤها من القيم الحرفية

```
let x = false;
```

new: ولكن يمكن أيضًا تعريف القيم المنطقية على أنها كائنات بالكلمة المحجوزة

```
let y = new Boolean(false);
```

(مثال ↓↓)

```
let x = false;  
let y = new Boolean(false);
```

```
// typeof x returns boolean  
// typeof y returns object
```


لا تتم بإنشاء كائنات منطقية

المحجوزة على تعقيد ال اكواد وإبطاء سرعة التنفيذ **new** تعمل الكلمة

:يمكن أن تنتج الكائنات المنطقية نتائج غير متوقعة

: متساويان **لا** و **X**، عند استخدام **==** المعامل

```
let x = false;  
let y = new Boolean(false);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

: غير متساويين **لا** و **X**، عند استخدام **===** المعامل

```
let x = false;  
let y = new Boolean(false);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

لاحظ الفرق بين $(x==y)$ و $(x===y)$.

صحيحة أو خاطئة؟ $(x == y)$

```
let x = new Boolean(false);  
let y = new Boolean(false);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

صحيحة أو خاطئة؟ $(x === y)$

```
let x = new Boolean(false);
```

```
let y = new Boolean(false);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

دائمًا ما تؤدي المقارنة بين كائنين في جافاسكريبت إلى ظهور خطأ

المقارنات

false أو **true** يتم استخدام عوامل المقارنة والمنطقية لاختبار

عوامل المقارنة

تُستخدم عوامل المقارنة في الكلمات المنطقية لتحديد المساواة أو الاختلاف بين المتغيرات أو القيم

:يوضح الجدول أدناه عوامل المقارنة، **x = 5** بالنظر إلى ذلك

Operator	وصف	Comparing	Returns
==	يساوي	x == 8	false
		x == 5	true
		x == "5"	true
===	قيمة متساوية ونوع متساو	x === 5	true
		x === "5"	false
!=	غير متساوي	x != 8	true
		x != 5	false
!==	قيمة غير متساوية أو نوع غير متساو	x !== "5"	true
		x !== 8	true
>	أكثر من	x > 8	false
<	أقل من	x < 8	true
>=	أكبر من أو يساوي	x >= 8	false
<=	اقل او يساوي	x <= 8	true

كيف يمكن استخدام عوامل المقارنة

يمكن استخدام عوامل المقارنة في العبارات الشرطية لمقارنة القيم واتخاذ الإجراءات بناءً على النتيجة

```
if (age < 18) abib = "Too young to buy alcohol";
```

سوف تتعلم المزيد عن استخدام العبارات الشرطية في الفصل التالي من هذا الكتاب

المعاملات المنطقية

يتم استخدام العوامل المنطقية لتحديد المنطق بين المتغيرات أو القيم

يشرح الجدول أدناه العوامل المنطقية. $x = 6$ و $y = 3$ بالنظر إلى ذلك

Operator	Description	مثال
&&	و	$(x < 10 \ \&\& \ y > 1)$ is true
	او	$(x == 5 \ \ y == 5)$ is false
!	ليس	$!(x == y)$ is true

العامل الشرطي (الثلاثي)

تحتوي جافاسكربت أيضًا على عامل شرطي يعين قيمة لمتغير بناءً على شرط ما

بناء الجملة

```
variableName = (condition) ? value1:value2
```

(مثال ✓ ↓↓)

```
let Habib = (age < 18) ? "Too young" : "Old enough";
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

إذا كان متغير العمر قيمة أقل من 18 عامًا، فإن قيمة المتغير القابل للتصويت ستكون "صغير جدًا"، وإلا فإن قيمة القابل للتصويت ستكون "كبير بما يكفي".

مقارنة الأنواع المختلفة

```
age = Number(age);
if (isNaN(age)) {
  Habib= "Input is not a number";
} else {
  Habib= (age < 18) ? "Too young" : "Old enough";
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

كيف استخدام عامل الدمج الفارغ (??)

(**undefined** أو **null**) يقوم عامل **??** التشغيل بارجاع الوسيطة الأولى إذا لم تكن فارغة. وإلا فإنه يقوم بارجاع الوسيطة الثانية.

(مثال ✓ ↓↓)

```
let Name = null;
let abib = "missing";
let result = Name ?? Habib;
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

مدعوم في جميع المتصفحات منذ مارس 2020 nullish عامل التشغيل

مشغل التسلسل الاختياري (.?)

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

(مثال ✓ ↓↓)

```
// Create an object:
const $Habib= {use:"Mohamed", age:"20", color:"white"};
// Ask for $Name:
document.getElementById("Habib").innerHTML = $Habib?.Name;
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

مشغل التسلسل الاختياري مدعوم في جميع المتصفحات منذ مارس 2020

كيف استخدام صيغة إذا، وإلا، وإلا إذا في البرمجة

نُستخدم العبارات الشرطية لتنفيذ إجراءات مختلفة بناءً على شروط مختلفة

عبارات شرطية

في كثير من الأحيان، عند كتابة ال اكواد ، تريد تنفيذ إجراءات مختلفة لقرارات مختلفة

يمكنك استخدام العبارات الشرطية في ال اكواد للقيام بذلك

في جافاسكربت لدينا العبارات الشرطية التالية

- لتحديد كتلة من ال اكواد المراد تنفيذها، إذا كان الشرط المحدد صحيحًا **if** يُستخدم
- لتحديد كتلة من ال اكواد المراد تنفيذها، إذا كان نفس الشرط خاطئًا **else** يُستخدم
- لتحديد شرط جديد للاختبار، إذا كان الشرط الأول خاطئًا **else if** يُستخدم
- لتحديد العديد من كتل ال اكواد البديلة التي سيتم تنفيذها **switch** يُستخدم

• ويرد كلمة الكلمة في الفصل التالي **switch**

كيف وضع شرط في الكود

العبرة لتحديد كتلة من تعليمات جافاسكربت البرمجية ليتم تنفيذها إذا كان الشرط صحيحًا **if** استخدم

بناء الجملة

```
if (condition) {
```

```
// block of code to be executed if the condition is true
```

```
}
```

إلى إنشاء خطأ جافاسكربت (IF أو IF) بالأحرف الصغيرة. ستؤدي الأحرف الكبيرة if لاحظ أنه

(مثال ✓)

قم بتحيةة "يوم جيد" إذا كانت الساعة أقل من 18:00

```
if (hour < 18) {  
    $Habib = "Good day";  
}
```

ستكون نتيجة التحية

```
//Good day
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

إذا الشرط صحيح افعل كذا وإلا افعل كذا

العبرة لتحديد كتلة من ال اكواد ليتم تنفيذها إذا كان الشرط وعبرة اخرى يتم تنفيذه اذا كان غير صحيح else استخدم

```
if (condition) {  
    // block of code to be executed if the condition is true  
}  
else {  
    // block of code to be executed if the condition is false  
}
```

(مثال ✓)

إذا كانت الساعة أقل من 18، قم بإنشاء تحية "يوم جيد"، وإلا "مساء الخير


```
if (hour < 18) {  
    $Habib = "Good day";  
} else {  
    $Habib = "Good evening";  
}
```

ستكون نتيجة التحيّة:

```
//Good evening
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

كلمة وإلا إذا

العبرة لتحديد شرط جديد إذا كان الشرط الأول خطأ **else if** استخدم

بناء الجملة

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false and condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false and condition2 is false  
}
```

(مثال ✓)

إذا كان الوقت أقل من 10:00، فقم بإنشاء تحية "صباح الخير"، وإذا لم يكن الأمر كذلك، ولكن الوقت أقل من 20:00، فقم بإنشاء تحية "يوم جيد"، وإلا "مساء الخير":

```
if (time < 10) {
  $Habib = "Good morning";
} else if (time < 20) {
  $Habib = "Good day";
} else {
  $Habib = "Good evening";
}
```

صيغة اخرى لوضع الشروط

لتنفيذ إجراءات مختلفة بناءً على شروط مختلفة **switch** يتم استخدام العبارة

استخدام الكلمة سيوتش

العبارة لتحديد إحدى كتل ال اكواد العديدة التي سيتم تنفيذها **switch** استخدم

بناء الجملة

```
switch(expression) {
```

```
case x:
```

```
// code block
```

```
break;
```

```
case y:
```

```
// code block
```

```
break;
```

```
default:
```



```
// code block
```

```
}
```

هذه هي الوظيفة التي يعمل بها

- يتم تقييم تعبير التبديل مرة واحدة.
- وتتم مقارنة قيمة الـ rameter مع قيم كل حالة.
- إذا كان هناك تطابق، فسيتم تنفيذ كتلة الـ code المرتبطة.
- إذا لم يكن هناك تطابق، فسيتم تنفيذ كتلة الـ code الافتراضية.

(مثال ✓ ↓↓)

بإرجاع أيام الأسبوع كرقم بين 0 و6 `getDay()` تقوم الوظيفة

(.الأحد=0، الاثنين=1، الثلاثاء=2)

يستخدم هذا الـ (مثال ✓ ↓↓) رقم يوم الأسبوع لحساب اسم يوم الأسبوع

```
switch (new Date().getDay()) {  
  case 0:  
    day = "Sunday";  
    break;  
  case 1:  
    day = "Monday";  
    break;  
  case 2:  
    day = "Tuesday";  
    break;  
  case 3:  
    day = "Wednesday";  
    break;  
  case 4:  
    day = "Thursday";  
    break;  
  case 5:  
    day = "Friday";  
    break;  
  case 6:
```

```
day = "Saturday";
```

```
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

الكلمة بريك

كلمة محجوزة ، فإنها تخرج من كتلة التبديل **break** عندما تصل جافاسكربت إلى

سيؤدي هذا إلى إيقاف التنفيذ داخل كتلة التبديل

ليس من الضروري كسر الحالة الأخيرة في كتلة التبديل. تنكسر الكتلة (تنتهي) هناك على أي حال

فسيتم تنفيذ الحالة التالية حتى لو لم يتطابق التقييم مع الحالة **Break**. ملاحظة: إذا قمت بحذف عبارة

وضع الشرط الافتراضي

المحجوزة الكود الذي سيتم تشغيله في حالة عدم وجود تطابق لحالة الأحرف **default** تحدد الكلمة

(مثال ↓↓↓)

بإرجاع أيام الأسبوع كرقم بين 0 و6 **getDay()** تقوم الوظيفة

إذا كان اليوم ليس السبت (6) ولا الأحد (0)، فاكتمل رسالة افتراضية

```
switch (new Date().getDay()) {  
  case 6:  
    abib = "Today is Saturday";  
    break;  
  case 0:  
    abib = "Today is Sunday";  
    break;  
  default:
```



```
abib = "Looking forward to the Weekend";  
}
```

نتيجة النص ستكون

Today is Sunday

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

هي الحالة الأخيرة في كتلة التبديل **default** لا يجب أن تكون الحالة

(مثال ✓) ↓↓

```
switch (new Date().getDay()) {  
  default:  
    abib = "Looking forward to the Weekend";  
    break;  
  case 6:  
    abib = "Today is Saturday";  
    break;  
  case 0:  
    abib = "Today is Sunday";  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

لم تكن الحالة الأخيرة في كتلة التبديل، فتذكر إنهاء الحالة الافتراضية بفاصل **default** إذا

كتل ال اكواد المشتركة

في بعض الأحيان قد ترغب في استخدام حالات تبديل مختلفة لنفس الرمز

في هذا ال (مثال ✓) ↓↓، تشترك الحالة 4 و5 في نفس كتلة ال اكواد ، وتشترك 0 و6 في كتلة تعليمات برمجية أخرى

(مثال ↓↓)

```
switch (new Date().getDay()) {  
  case 4:  
  case 5:  
    abib = "Soon it is Weekend";  
    break;  
  case 0:  
  case 6:  
    abib = "It is Weekend";  
    break;  
  default:  
    abib = "Looking forward to the Weekend";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

تفاصيل التبديل

إذا تطابقت حالات متعددة مع قيمة حالة، فسيتم تحديد الحالة الأولى.

إذا لم يتم العثور على أي حالات مطابقة، فسيستمر البرنامج في التسمية الافتراضية.

إذا لم يتم العثور على تسمية افتراضية، فسيستمر البرنامج في العبارة (العبارات) بعد التبديل.

المقارنة الصارمة

(===) تستخدم حالات التبديل مقارنة صارمة.

يجب أن تكون القيم من نفس النوع للمطابقة.

لا يمكن أن تكون المقارنة الصارمة صحيحة إلا إذا كانت المعاملات من نفس النوع.

x: في هذا ال (مثال ↓↓) لن يكون هناك تطابق لـ

(مثال ↓↓)

```
let x = "0";  
switch (x) {  
  case 0:  
    abib = "Off";  
    break;
```



```
case 1:
  abib = "On";
  break;
default:
  abib = "No value found";
}
```

الحلقة

يمكن للحلقات تنفيذ كتلة من ال اكواد عدة مرات

مثال

تعتبر الحلقات مفيدة إذا كنت تريد تشغيل نفس الكود مرارًا وتكرارًا، وفي كل مرة بقيمة مختلفة

غالبًا ما يكون هذا هو الحال عند العمل مع المصفوفات

بدلاً من الكتابة:

```
abib += HaBiB[0] + "<br>";
abib += HaBiB[1] + "<br>";
abib += HaBiB[2] + "<br>";
abib += HaBiB[3] + "<br>";
abib += HaBiB[4] + "<br>";
abib += HaBiB[5] + "<br>";
```

يمكنك كتابة:

```
for (let i = 0; i < HaBiB.length; i++) {
  abib += HaBiB[i] + "<br>";
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

أنواع مختلفة من الحلقات

تدعم جافاسكربت أنواعًا مختلفة من الحلقات:

- **for** - حلقات من خلال كتلة من ال اكواد عدة مرات
- **for/in** - حلقات من خلال خصائص الكائن
- **for/of** - حلقات من خلال قيم كائن قابل للتكرار
- **while** - حلقات من خلال كتلة من ال اكواد عندما يكون الشرط المحدد صحيحًا
- **do/while** - يتكرر أيضًا خلال كتلة من ال اكواد عندما يكون الشرط المحدد صحيحًا

For الحلقة

حلقة تحتوي على 3 تعبيرات اختيارية **for** تنشئ العبارة:

```
for (expression 1; expression 2; expression 3) {
```

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

```
}
```

- يتم تنفيذ الرامتر 1 (مرة واحدة) قبل تنفيذ كتلة ال اكواد
- يحدد الرامتر 2 شرط تنفيذ كتلة ال اكواد
- يتم تنفيذ الرامتر 3 (في كل مرة) بعد تنفيذ مقطع ال اكواد

(مثال ↓↓) ✓

```
for (let i = 0; i < 5; i++) {  
  abib += "The number is " + i + "<br>";  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

نومن ال (✓ مثال ↓↓) أعلاه يمكنك أن تقرأ

($i = 0$) يقوم الرامتر 1 بتعيين متغير قبل بدء الحلقة

(أقل من 5 i يجب أن يكون) يحدد الرامتر 2 شرط تشغيل الحلقة

في كل مرة يتم فيها تنفيذ كتلة ال اكواد في الحلقة ($i++$) يزيد الرامتر 3 القيمة

البرامتر 1

عادةً ستستخدم البرامتر 1 لتهيئة المتغير المستخدم في الحلقة.
هذا ليس هو الحال دائماً. جافاسكربت لا يهتم. البرامتر 1 اختياري.
يمكنك بدء العديد من القيم في البرامتر 1 (مفصولة بفاصلة)

(↓↓↓ مثال ✓)

```
for (let i = 0, len = HaBiB.length, abib = ""; i < len; i++) {  
  abib += HaBiB[i] + "<br>";  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

ويمكنك حذف البرامتر 1 (مثلما يتم تعيين القيم قبل بدء الحلقة)

(↓↓↓ مثال ✓)

```
let i = 2;  
let len = HaBiB.length;  
let abib = "";  
for (; i < len; i++) {  
  abib += HaBiB[i] + "<br>";  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

البرامتر 2

غالبًا ما يتم استخدام البرامتر 2 لتقييم حالة المتغير الأولي.
هذا ليس هو الحال دائماً. جافاسكربت لا يهتم. البرامتر 2 اختياري أيضًا.
إذا عاد البرامتر 2 إلى القيمة الصحيحة، فستبدأ الحلقة من جديد. إذا عادت خطأ، ستنتهي الحلقة.

إذا قمت بحذف البرامتر 2، فيجب عليك توفير فاصل داخل الحلقة. وإلا فإن الحلقة لن تنتهي أبداً. سيؤدي هذا إلى تعطل المتصفح. اقرأ عن فترات الايقاف في فصل لاحق من هذا الكتاب.

البرامتر 3

غالبًا ما يزيد البرامتر 3 قيمة المتغير الأولي.

هذا ليس هو الحال دائمًا. جافاسكربت لا يهتم. البرامتر 3 اختياري.

أو أي شيء آخر، $(i = i + 15)$ أو الزيادة الموجبة، $(i--)$ يمكن للتعبير 3 أن يفعل أي شيء مثل الزيادة السالبة.

يمكن أيضًا حذف البرامتر 3 (مثلًا عند زيادة قيمك داخل الحلقة)

(مثال ✓ ↓↓)

```
let i = 0;
let len = HaBiB.length;
let abib = "";
for (; i < len; ) {
  abib += HaBiB[i] + "<br>";
  i++;
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

نطاق الحلقة

في حلقة **var** باستخدام

(مثال ✓ ↓↓)

```
var i = 5;

for (var i = 0; i < 10; i++) {
  // some code
}

// Here i is 10
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

في حلقة **let** باستخدام

(مثال ↓↓↓)

```
let i = 5;

for (let i = 0; i < 10; i++) {
  // some code
}

// Here i is 5
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

يقوم المتغير المعلن في الحلقة بإعادة تعريف المتغير الموجود خارج الحلقة، **var** في ال (مثال ↓↓↓) الأول، باستخدام لا يقوم المتغير المعلن في الحلقة بإعادة تعريف المتغير خارج الحلقة، **let** في ال (مثال ↓↓↓) الثاني، باستخدام مرئيًا فقط داخل الحلقة في حلقة، سيكون المتغير **let** استخدامه للإعلان عن المتغير **let** عند

في الفصل التالي **for/of** والحلقة **for/in** سيتم شرح الحلقة

حلقة بينما

في الفصول التالية **the do/while** و **while** سيتم شرح الحلقة

For In حلقة

عبر خصائص الكائن **for in** يتم تنفيذ عبارة جافاسكربت

بناء الجملة

```
for (key in object) {
  // code block to be executed
}
```

(مثال ↓↓↓)

```
const hAbiB = {fName:"Habib", lName:"Al Husini 😊😊😊", age:18};

let abib = "";
```

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

مثال آخر

- على كائن عنصر **for in** تتكرر حلقة
- **(x)** كل تكرار يقوم بإرجاع مفتاح
- يتم استخدام المفتاح للوصول إلى قيمة المفتاح
- **[x]** قيمة المفتاح هي العنصر

التنفيذ على المصفوفة

فوق خصائص المصفوفة **for in** يمكن أيضًا تنفيذ عبارة جافاسكريبت

بناء الجملة

```
for (variable in array) {  
  code  
}
```

(↓ ↓) مثال (✓)

```
const abibNum = [45, 4, 9, 16, 25];
```

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

إذا كان ترتيب الفهرس مهمًا **for in over a Array** لا تستخدم

يعتمد ترتيب الفهرس على التنفيذ، وقد لا يمكن الوصول إلى قيم ال مصفوفة بالترتيب الذي تتوقعه

عندما يكون الترتيب مهمًا (`Array.forEach()` أو `for` ، أو حلقة `for` من الأفضل استخدام حلقة

() forEach على المصفوفة

دالة (دالة رد اتصال) مرة واحدة لكل عنصر من عناصر المصفوفة (`forEach()`) تستدعي الوظيفة

(مثال ✓) ↓↓

```
const abibNum = [45, 4, 9, 16, 25, 45, 4, 9, 16, 25];  
  
let hAbiB = "";  
abibNum.forEach(Husini);  
  
function Husini(value, index, array) {  
  hAbiB += value;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

ملاحظ أن الدالة تأخذ 3 وسيطات

- قيمة العنصر
- فهرس البند
- المصفوفة نفسها

يستخدم ال (مثال ✓) ↓↓ أعلاه معلمة القيمة فقط. يمكن إعادة كتابته إلى

(مثال ✓) ↓↓

```
const abibNum = [45, 4, 9, 16, 25];  
  
let hAbiB = "";  
abibNum.forEach(Husini);  
  
function Husini(value) {  
  hAbiB += value;  
}
```

تابع انواع الحلقات

عبر قيم كائن قابل للتكرار `for of` يتم تنفيذ عبارة جافاسكريبت

فهو يتيح لك التكرار عبر هياكل الكلمات القابلة للتكرار مثل المصفوفات والنصوص والخرائط وقوائم العقد والمزيد

بناء الجملة

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

var أو **let** بـ **const** متغير - لكل تكرار يتم تعيين قيمة الخاصية التالية للمتغير. يمكن الإعلان عن المتغير قابل للتكرار - كائن له خصائص قابلة للتكرار.

التكرار فوق مصفوفة

(مثال ↓↓↓)

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

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

التكرار على نص

(مثال ↓↓↓)

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

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

في الفصل التالي **do/while** والحلقة **while** سيتم شرح الحلقة

أثناء الحلقة

يمكن للحلقات تنفيذ كتلة من ال اكواد طالما كان الشرط المحدد صحيحًا

"حلقة" بينما

عبر كتلة من ال اكواد طالما كان الشرط المحدد صحيحًا **while** تتكرر الحلقة

بناء الجملة

```
while (condition) {
```

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

```
}
```

(مثال ✓)

أقل من 10 (i) في ال (مثال ✓) التالي، سيتم تشغيل ال اكواد الموجودة في الحلقة مرًا وتكرًا، طالما أن المتغير

(مثال ✓)

```
while (i < 10) {  
  abib += "The number is " + i;  
  i++;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

إذا نسيت زيادة المتغير المستخدم في الشرط، فلن تنتهي الحلقة أبدًا. سيؤدي هذا إلى تعطل المتصفح

"حلقة" افعل أثناء كذا.

ستنفيذ هذه الحلقة كتلة ال اكواد مرة واحدة، قبل التحقق مما إذا كان **while** هي نسخة مختلفة من الحلقة **do while** الحلقة الشرط صحيحًا، ثم تكرر الحلقة طالما كان الشرط صحيحًا.

بناء الجملة

```
do {
```

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

```
}
```

```
while (condition);
```

(↓↓↓) مثال (✓)

حلقة. سيتم دائمًا تنفيذ الحلقة مرة واحدة على الأقل، حتى لو كان الشرط خاطئًا، **do while** يستخدم ال (✓) مثال (↓↓↓) أدناه لأنه يتم تنفيذ كتلة ال اكواد قبل اختبار الشرط

(↓↓↓) مثال (✓)

```
do {  
  abib += "The number is " + i;  
  i++;  
}  
while (i <= 10);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

لا تنس زيادة المتغير المستخدم في الشرط، وإلا فلن تنتهي الحلقة أبدًا

مقارنة لحلقة بينما

مع حذف **for**، تشبه إلى حد كبير حلقة **while** فسوف تكتشف أن حلقة **for**، إذا كنت قد قرأت الفصل السابق، حول حلقة العبرة 1 والعبرة 3.

: حلقة لجمع أسماء من مصفوفة **for** تستخدم الحلقة في هذا ال (✓) مثال (↓↓↓)

(↓↓↓ مثال ✓)

```
const HaBiB= ["Abu Habib Al-Husini", "Hamza", "Habib", "Habib"];
let i = 0;
let abib_ = "";

for (;HaBiB[i];) {
  abib_ += HaBiB[i];
  i++;
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

: حلقة لجمع أسماء من مصفوفة **while** تستخدم الحلقة في هذا ال (↓↓↓ مثال ✓)

(↓↓↓ مثال ✓)

```
const HaBiB= ["Abu Habib Al-Husini", "Hamza", "Habib", "Habib"];
let i = 0;
let abib = "";

while (HaBiB[i]) {
  abib += HaBiB[i];
  i++;
}
```

تخرج " من الحلقة **break** العبارة

تقفز فوق " تكرر واحد في الحلقة **continue** العبارة

كلمة التوقف من داخل الحلقة

الكلمة **switch()** العبارة المستخدمة في فصل سابق من هذا الكتاب. تم استخدامه "للقفز" من **break** لقد رأيت بالفعل

:لانتقال من الحلقة **break** يمكن أيضًا استخدام العبارة

(↓↓↓ مثال ✓)

```
for (let i = 0; i < 10; i++) {
  if (i === 3) { break; }
  abib += "The number is " + i + "<br>";
}
```

هو 3 (i) تنهي العبارة الحلقة ("تكسر" الحلقة) عندما يكون عداد الحلقة **break**، في ال (✓ مثال ↓↓) أعلاه

كامة الاستمرار

تكررًا واحدًا (في الحلقة)، في حالة حدوث شرط محدد، وتستمر مع التكرار التالي في الحلقة **continue** تكسر العبارة

:يتخطى هذا ال (✓ مثال ↓↓) قيمة 3

(✓ مثال ↓↓)

```
for (let i = 0; i < 10; i++) {  
  if (i === 3) { continue; }  
  abib += "The number is " + i + "<br>";  
}
```

تسميات

:لتسمية عبارات جافاسكريبت، عليك أن تسبق العبارات باسم تسمية ونقطتين

```
label:  
statements
```

هي عبارات جافاسكريبت الوحيدة التي يمكنها "القفز من فوق عنصر او خارج" الحلقة ال **break** و **continue** إن عبارات
اكواد .

:بناء الجملة

```
break labelName;
```

```
continue labelName;
```

. إلا لتخطي تكرار حلقة واحدة (مع أو بدون مرجع تسمية) **continue** لا يمكن استخدام العبارة

. التي لا تحتوي على مرجع تسمية، إلا للانتقال من حلقة أو مفتاح تبديل، **break** لا يمكن استخدام العبارة

: للانتقال من أي كتلة تعليمات برمجية **Break** باستخدام مرجع التسمية، يمكن استخدام عبارة

(↓↓ مثال ✓)

```
const HaBiB= ["Abu Habib Al-Husini", "Hamza", "Habib", "Habib"];
list: {
  abib += HaBiB[0] + "<br>";
  abib += HaBiB[1] + "<br>";
  break list;
  abib += HaBiB[2] + "<br>";
  abib += HaBiB[3] + "<br>";
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

كتلة ال اكواد هي كتلة من ال اكواد بين { و }

عناصر التكرارية

العناصر القابلة للتكرار هي كائنات قابلة للتكرار (مثل المصفوفات)

يمكن الوصول إلى العناصر القابلة للتكرار باستخدام تعليمات برمجية بسيطة وفعالة

الحلقات **for..of** يمكن تكرار العناصر التكرارية باستخدام

للحلقه

عبر عناصر الكائن القابل للتكرار **for..of** تتكرر عبارة جافاسكريبت

بناء الجملة

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

التكرار

التكرار سهل الفهم.

إنه يعني ببساطة التكرار على نص من العناصر.

فيما يلي بعض الأمثلة السهلة:

- التكرار على نص
- التكرار على مصفوفة

التكرار على نص

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

(مثال ↓↓↓)

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

التكرار على مصفوفة

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

(مثال ↓↓↓)

```
const Husinie = ["a","b","c"];  
  
for (const x of Husinie) {  
  // code block to be executed  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

التكرار على مجموعة

حلقة للتكرار على عناصر المجموعة **for..of** يمكنك استخدام

(مثال ✓ ↓↓)

```
const Husinie = new Set(["a","b","c"]);  
  
for (const x of Husinie) {  
  // code block to be executed  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

التكرار على خريطة المصفوفه

حلقة للتكرار على عناصر خريطة المصفوف **for..of** يمكنك استخدام

(مثال ✓ ↓↓)

```
const HaBiB= new Map([  
  ["Abo Habip", 500],  
  ["Al-Husini__", 300],  
  ["Osman", 200]  
]);  
  
for (const x of HaBiB) {  
  // code block to be executed  
}
```

مجموعات

مجموعة جافاسكربت عبارة عن مجموعة من القيم الفريدة.

يمكن أن تحدث كل قيمة مرة واحدة فقط في المجموعة.

طرق المجموعة المحبوزة

Method	وصف
<code>new Set()</code>	إنشاء مجموعة جديدة
<code>add()</code>	يضيف عنصرًا جديدًا إلى المجموعة
<code>delete()</code>	إزالة عنصر من مجموعة
<code>has()</code>	يُرجع صحيحًا في حالة وجود قيمة في المجموعة
<code>forEach()</code>	يستدعي رد اتصال لكل عنصر في المجموعة
<code>values()</code>	إرجاع مكرر بجميع القيم الموجودة في المجموعة
<code>size</code>	إرجاع عدد العناصر 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

Set () وظيفة

:المنشئ `new Set()` تمرير مصفوفة إلى

(مثال ↓↓) ✓

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

إنشاء مجموعة وإضافة القيم

(مثال ↓↓) ✓

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

// Add Values to the Set
Husinie.add("a");
Husinie.add("b");
Husinie.add("c");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

إنشاء مجموعة وإضافة المتغيرات

(مثال ↓↓) ✓

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

// Create Variables
const a = "a";
const b = "b";
const c = "c";

// Add Variables to the Set
Husinie.add(a);
Husinie.add(b);
Husinie.add(c);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

وظيفة الإضافة

(مثال ✓ ↓↓)

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

إذا أضفت عناصر متساوية، فسيتم حفظ العنصر الأول فقط

(مثال ✓ ↓↓)

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

forEach() وظيفة

ندالة لكل عنصر مجموعة **forEach()** تستدعي الوظيفة

(مثال ✓ ↓↓)

```
// Create a Set  
const Husinie = new Set(["a", "b", "c"]);  
  
// List all Elements  
let abib = "";  
Husinie.forEach (function(value) {  
  abib+= value;  
})
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

values() وظيفة

تقوم الوظيفة بإرجاع كائن مكرر جديد يحتوي على جميع القيم الموجودة في المجموعة

(مثال ↓↓)

Husnie.values() // Returns [object Set Iterator]

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

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

(مثال ↓↓)

```
// List all Elements
let abib = "";
for (const x of Husnie.values()) {
  abib += x;
}
```

خرائط

تحتوي خريطة المصفوف على أزواج قيمة المفتاح حيث يمكن أن تكون المفاتيح أي نوع كلمات

تتذكر خريطة المصفوف ترتيب الإدراج الأصلي للمفاتيح

كيفية إنشاء خريطة

يمكنك إنشاء خريطة جافاسكريبت عن طريق

- **new Map()** تمرير مصفوفة إلى
- **Map.set()** إنشاء خريطة واستخدامها

new Map() وظيفة

: يمكنك إنشاء خريطة عن طريق تمرير مصفوفة إلى المُنشئ

(مثال ✓) ↓↓

```
// Create a Map
const HaBiB= new Map([
  ["Abo Habip", 500],
  ["Al-Husini__", 300],
  ["Osman", 200]
]);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

set() وظيفة

: يمكنك إضافة عناصر إلى خريطة المصفوف باستخدام الوظيفة

(مثال ✓) ↓↓

```
// Create a Map
const HaBiB= new Map();

// Set Map Values
HaBiB.set("Abo Habip", 500);
HaBiB.set("Al-Husini__", 300);
HaBiB.set("Osman", 200);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

:لتغيير قيم خريطة المصفوف الموجودة set() يمكن أيضاً استخدام الوظيفة

(مثال ✓) ↓↓

```
HaBiB.set("Abo Habip", 200);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

وظيفة الحصول على قيمة

: على قيمة المفتاح في خريطة المصفوف `get()` تحصل الوظيفة

(مثال ✓) ↓↓

```
HaBiB.get("Abo Habip"); // Returns 500
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

خاصية الحجم

: بإرجاع عدد العناصر في خريطة المصفوف `size` تقوم الخاصية

(مثال ✓) ↓↓

```
HaBiB.size;
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

وظيفة الحذف

: بإزالة عنصر خريطة المصفوف `delete()` تقوم الوظيفة

(مثال ✓) ↓↓

```
HaBiB.delete("Abo Habip");
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

الوظيفة `has()`

: صحيحًا في حالة وجود مفتاح في خريطة المصفوف `has()` تُرجع الوظيفة

(↓↓↓ مثال ✓)

```
HaBiB.has("Abo Habip");
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

جرب هذا:

```
HaBiB.delete("Abo Habip");
```

```
HaBiB.has("Abo Habip");
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

كائنات مقارنة بالخرائط

الاختلافات بين كائنات جافاسكريبت والخرائط:

خريطة	هدف
قابلة للتكرار مباشرة	غير قابلة للتكرار بشكل مباشر
لديك خاصية الحجم	ليس لديك خاصية الحجم
يمكن أن تكون المفاتيح أي نوع كلمات	يجب أن تكون المفاتيح عبارة عن نصوص (أو اكواد)
يتم ترتيب المفاتيح عن طريق الإدراج	المفاتيح ليست مرتبة بشكل جيد
ليس لديك مفاتيح افتراضية	لديك مفاتيح افتراضية

forEach() وظيفة

دالة لكل زوج من المفاتيح/القيمة في خريطة المصفوف (forEach()) تستدعي الوظيفة

(↓↓↓ مثال ✓)

```
// List all entries
let abib = {};
HaBiB.forEach(function(value, key) {
  abib += key + ' = ' + value;
})
```


وظيفة الإدخالات

: بإرجاع كائن مكرر مع [مفتاح العنصر] في خريطة المصفوف (**entries()**) تقوم الوظيفة

(مثال ↓↓↓)

```
// List all entries
let abib = {};
for (const x of HaBiB.entries()) {
  abib += x;
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

: يوجد في جافاسكربت 5 أنواع مختلفة من الكلمات التي يمكن أن تحتوي على قيم

- string
- number
- boolean
- object
- function

: هناك 6 أنواع من الكائنات

- Object
- Date
- Array
- String
- Number
- Boolean

نوعان من الكلمات لا يمكن أن يحتوي على قيم

- **null**
- **undefined**

نوع المشغل

عامل التشغيل للعثور على نوع كلمات متغير جافاسكربت **typeof** يمكنك استخدام

(مثال ↓↓↓)

```
typeof "Habib" // Returns "string"
typeof 3.14 // Returns "number"
typeof NaN // Returns "number"
typeof false // Returns "boolean"
typeof [1,2,3,4] // Returns "object"
typeof {Name:'Habib', age:34} // Returns "object"
typeof new Date() // Returns "object"
typeof function () {} // Returns "function"
typeof my$Habib // Returns "undefined" *
typeof null // Returns "object"
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

يرجى ملاحظة ما يلي:

- هو رقم **NaN** نوع كلمات
- نوع كلمات المصفوفة هو كائن
- نوع كلمات التاريخ هو كائن
- نوع الكلمات الفارغة هو كائن
- * نوع كلمات المتغير غير المحدد غير محدد
- * نوع كلمات المتغير الذي لم يتم تعيين قيمة له غير محدد أيضًا

لتحديد ما إذا كان كائن جافاسكربت عبارة عن مصفوفة (أو تاريخ) **typeof** لا يمكنك استخدامه

الكلمات البدائية

قيمة الكلمات البدائية هي قيمة كلمات بسيطة واحدة بدون خصائص وطرق إضافية

إرجاع أحد هذه الأنواع البدائية **typeof** يمكن للمشغل

- string
- number
- boolean
- undefined

(مثال ↓↓↓)

```
typeof "Habib" // Returns "string"  
typeof 3.14 // Returns "number"  
typeof true // Returns "boolean"  
typeof false // Returns "boolean"  
typeof x // Returns "undefined" (if x has no value)
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

ايجاد النوع

إرجاع أحد النوعين **typeof** يمكن للمشغل

- function
- object

التشغيل بإرجاع "كائن" للكائنات والمصفوفات والقيمة الخالية **typeof** يقوم عامل

التشغيل بإرجاع "كائن" للوظائف **typeof** لا يقوم عامل

(مثال ↓↓↓)

```
typeof {Name:'Habib', age:34} // Returns "object"  
typeof [1,2,3,4] // Returns "object" (not "array", see note below)  
typeof null // Returns "object"  
typeof function myFunc(){ } // Returns "function"
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

للمصفوفات لأن المصفوفات في جافاسكريبت عبارة عن كائنات "object" التشغيل بإرجاع **typeof** يقوم عامل

تابع typeof

ليس متغيراً، إنه عامل. المشغلون (+ - * /) ليس لديهم أي نوع كلمات **typeof** المشغل. عامل التشغيل يقوم دائماً بإرجاع نص (تحتوي على نوع المعامل) **typeof** لكن

خاصية البناء

إرجاع وظيفة المنشئ لجميع متغيرات جافاسكربت **constructor** تقوم الخاصية

(مثال ↓↓) ✓

```
"Habib".constructor // Returns function String() {[native code]}
(3.14).constructor // Returns function Number() {[native code]}
false.constructor // Returns function Boolean() {[native code]}
[1,2,3,4].constructor // Returns function Array() {[native code]}
{Name:'Habib',age:34}.constructor // Returns function Object() {[native code]}
new Date().constructor // Returns function Date() {[native code]}
function () {}.constructor // Returns function Function() {[native code]}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

("يحتوي على كلمة "مصفوفة") **Array** يمكنك التحقق من خاصية المنشئ لمعرفة ما إذا كان الكائن

(مثال ↓↓) ✓

```
function isArray(Husnie_Array) {
  return Husnie_Array.constructor.toString().indexOf("Array") > -1;
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

Array : أو حتى بشكل أبسط، يمكنك التحقق مما إذا كان الكائن عبارة عن دالة

(مثال ↓↓) ✓

```
function isArray(Husnie_Array) {
  return Husnie_Array.constructor === Array;
}
```


كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

:"يحتوي على كلمة "تاريخ" Date يمكنك التحقق من خاصية المنشئ لمعرفة ما إذا كان الكائن

(مثال ✓ ↓↓)

```
function isDate(myDate) {  
  return myDate.constructor.toString().indexOf("Date") > -1;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

: أو حتى بشكل أبسط، يمكنك التحقق مما إذا كان الكائن عبارة عن دالة تاريخ

(مثال ✓ ↓↓)

```
function isDate(myDate) {  
  return myDate.constructor === Date;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

غير معرف

. **undefine** في جافاسكريبت، المتغير بدون قيمة له القيمة . النوع هو أيضا

(مثال ✓ ↓↓)

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

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

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

(مثال ✓ ↓↓)

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

القيم الفارغة

undefine. القيمة الفارغة ليس لها علاقة بـ
تحتوي النص الفارغة على قيمة قانونية ونوع

(مثال ↓↓)

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

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

خطا

يوجد "لا شيء". من المفترض أن يكون شيئاً غير موجود **null** في جافاسكريبت
هو كائن **null** لسوء الحظ، في جافاسكريبت، نوع الكلمات

null كائن. ينبغي أن يكون **typeof null** يمكنك اعتباره خطأ في جافاسكريبت وهو

null: يمكنك إفراغ كائن عن طريق تعيينه على

(مثال ↓↓)

```
let hAbiB = {firstName:"Habib", lastName:"Al Husini 🇸🇩🇸🇩🇸🇩", age:50, eyeColor:"blue"};  
hAbiB = null; // Now value is null, but type is still an object
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

undefined: يمكنك أيضاً إفراغ كائن عن طريق تعيينه على

(مثال ↓↓)

```
let hAbiB = {firstName:"Habib", lastName:"Al Husini 🇸🇩🇸🇩🇸🇩", age:50, eyeColor:"blue"};  
hAbiB = undefined; // Now both value and type is undefined
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

الفرق بين غير محدد وفارغ

في القيمة ومختلفتان في النوع **null** ومتساويتان **undefined**:

```
typeof undefined // undefined
typeof null      // object

null === undefined // false
null == undefined // true
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

مثيل المشغل

كان الكائن مثيلاً للكائن المحدد **true** يُرجع عامل التشغيل إذا **instanceof**:

(مثال ↓↓↓)

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

(HaBiB instanceof Array);
(HaBiB instanceof Object);
(HaBiB instanceof String);
(HaBiB instanceof Number);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

المشغل الفارغ

بتقييم اليرامتر وإرجاع غير محدد . غالباً ما يُستخدم هذا العامل للحصول على القيمة الأولية غير **void** يقوم عامل التشغيل (مفيد عند تقييم تعبير دون استخدام القيمة المرجعة) **"void(0)"** المحددة، باستخدام

(مثال ↓↓↓)

```
<a href="javascript:void(0);">
  Useless link
</a>
```

``
Click me to change the background color of body to red
``

تحويل نوع

يمكن تحويل متغيرات جافاسكربت إلى متغير جديد ونوع كلمات آخر

- عن طريق استخدام وظيفة جافاسكربت
- تلقائياً عن طريق جافاسكربت نفسها

تحويل النصوص إلى أرقام

بتحويل متغير (أو قيمة) إلى رقم `Number()` تقوم الوظيفة العامة

يتم تحويل نص رقمية (مثل "3.14") إلى رقم (مثل 3.14)

يتم تحويل نص فارغة (مثل "") إلى 0

(ليس رقمًا) `NaN` إلى ("Habib") مثل) يتم تحويل نص غير رقمية

أمثلة

سيتم تحويل هذه

```
Number("3.14")
```

```
Number(Math.PI)
```

```
Number(" ")
```

```
Number("")
```

لن يتم تحويل هذه

```
Number("99 88")
```

```
Number("Habib")
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

طرق ووظائف الأرقام

ستجد في الفصل طرق الأرقام المزيد من الطرق التي يمكن استخدامها لتحويل النصوص إلى أرقام:

Method	وصف
Number()	إرجاع رقم، تم تحويله من الوسيط الخاص به
parseFloat()	ياخذ نص ويعيد رقم الفاصلة العشرية
parseInt()	ياخذ نص ويعيد عددا صحيحا

المشغل الأحادي +

يمكن استخدام العامل الأحادي + لتحويل المتغير إلى رقم: للتذكير

(مثال ✓ ↓↓)

```
let y = "5"; // y is a string
let x = + y; // x is a number
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

(ليس رقماً) NaN إذا تعذر تحويل المتغير، فسيظل رقماً، ولكن بقيمة

(مثال ✓ ↓↓)

```
let y = "Habib"; // y is a string
let x = + y; // x is a number (NaN)
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

كيف تحويل الأرقام إلى نصوص

تحويل الأرقام إلى نصوص (`String()`) يمكن للوظيفة العامة

يمكن استخدامه على أي نوع من الأرقام أو القيم الحرفية أو المتغيرات أو اليوامترات

(مثال ↓↓) ✓

```
String(x) // returns a string from a number variable x
String(123) // returns a string from a number literal 123
String(100 + 23) // returns a string from a number from an expression
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

تفعل الشيء نفسه `toString()` وظيفة الرقم

(مثال ↓↓) ✓

```
x.toString()
(123).toString()
(100 + 23).toString()
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

المزيد من الدوال

ستجد في الفصل طرق الأرقام المزيد من الطرق التي يمكن استخدامها لتحويل الأرقام إلى نصوص في كتاب المرجع الكامل للجافاسكريبت

Method	وصف
<code>toExponential()</code>	إرجاع نص برقم مقرب ومكتوب باستخدام التدوين الأسّي
<code>toFixed()</code>	تُرجع نص تحتوي على رقم مقرب ومكتوب بعدد محدد من الكسور العشرية
<code>toPrecision()</code>	إرجاع نص برقم مكتوب بطول محدد

تحويل التواريخ إلى أرقام

لتحويل التواريخ إلى أرقام `Number()` يمكن استخدام الوظيفة العامة

```
d = new Date();
Number(d) // returns 1404568027739
```

تفعل الشيء نفسه `getTime()` وظيفة التاريخ


```
d = new Date();
abibDate.getTime() // returns 1404568027739
```

تحويل التواريخ إلى نصوص

تحويل التواريخ إلى نصوص (`String()`) يمكن للوظيفة العامة

```
String(Date()) // returns "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)"
```

تفعل الشيء نفسه (`toString()`) ووظيفة التاريخ

(مثال ✓ ↓↓)

```
Date().toString() // returns "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)"
```

ستجد في فصل طرق التاريخ المزيد من الطرق التي يمكن استخدامها لتحويل التواريخ إلى نصوص:

Method	وصف
<code>getDate()</code>	احصل على اليوم كرقم (1-31)
<code>getDay()</code>	احصل على رقم أيام الأسبوع (0-6)
<code>getFullYear()</code>	احصل على السنة المكونة من أربعة أرقام (yyyy)
<code>getHours()</code>	احصل على الساعة (0-23)
<code>getMilliseconds()</code>	احصل على المللي ثانية (0-999)
<code>getMinutes()</code>	احصل على الدقائق (0-59)
<code>getMonth()</code>	احصل على الشهر (0-11)
<code>getSeconds()</code>	احصل على الثواني (0-59)
<code>getTime()</code>	احصل على الوقت (ملي ثانية منذ 1 يناير 1970)

تحويل القيم المنطقية إلى أرقام

أيضاً تحويل القيم المنطقية إلى أرقام (`Number()`) يمكن للوظيفة العامة

```
Number(false) // returns 0
```

```
Number(true) // returns 1
```

تحويل القيم المنطقية إلى نصوص

تحويل القيم المنطقية إلى نصوص (`String()`) يمكن للوظيفة العامة

```
String(false) // returns "false"
```

```
String(true) // returns "true"
```

تفعل الشيء نفسه (`toString()`) الوظيفة المنطقية

```
false.toString() // returns "false"
```

```
true.toString() // returns "true"
```

التحويل التلقائي للنوع

"عندما تحاول جافاسكريبت العمل على نوع كلمات "خاطئ"، ستحاول تحويل القيمة إلى نوع "صحيح

والنتيجة ليست دائماً ما تتوقعه

```
5 + null // returns 5 because null is converted to 0
```

```
"5" + null // returns "5null" because null is converted to "null"
```

```
"5" + 2 // returns "52" because 2 is converted to "2"
```

```
"5" - 2 // returns 3 because "5" is converted to 5
```

```
"5" * "2" // returns 10 because "5" and "2" are converted to 5 and 2
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

تحويل النص التلقائي

عندما تحاول "إخراج" كائن أو متغير (`toString()`) تقوم جافاسكريبت تلقائياً باستدعاء وظيفة المتغير

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

```
// if abibVar= {Name:"FHabib"} // toString converts to "[object Object]"
```

```
// if abibVar= [1,2,3,4] // toString converts to "1,2,3,4"
```



```
// if abibVar= new Date() // toString converts to "Fri Jul 18 2014 09:08:55  
GMT+0200"
```

يتم أيضًا تحويل الأرقام والقيم المنطقية، لكن هذا غير مرئي تمامًا

```
// if abibVar= 123 // toString converts to "123"  
// if abibVar= true // toString converts to "true"  
// if abibVar= false // toString converts to "false"
```

جدول تحويل الأنواع في

Boolean و String و Number يوضح هذا الجدول نتيجة تحويل قيم جافاسكربت المختلفة إلى

القيمة	التحويل الى رقم	التحويل الى نص	التحويل الى بولين
false	0	"false"	false
true	1	"true"	true
0	0	"0"	false
1	1	"1"	true
"0"	0	"0"	true
"000"	0	"000"	true
"1"	1	"1"	true
NaN	NaN	"NaN"	false
Infinity	Infinity	"Infinity"	true
-Infinity	-Infinity	"-Infinity"	true
""	0	""	false
"20"	20	"20"	true
"twenty"	NaN	"twenty"	true
[]	0	""	true
[20]	20	"20"	true
[10,20]	NaN	"10,20"	true
["twenty"]	NaN	"twenty"	true
["ten","twenty"]	NaN	"ten,twenty"	true
function(){}	NaN	"function(){}"	true

{}	NaN	"[object Object]"	true
null	0	"null"	false
undefined	NaN	"undefined"	false

• تشير القيم الموجودة بين علامتي الاقتباس إلى قيم النص.

• تشير القيم الحمراء إلى قيم (بعض) قد لا يتوقعها المبرمجون.

Bitwise عمليات جافاسكربت

Bitwise مشغلي

المعامل	الاسم	وصف
&	AND	يضبط كل بت على 1 إذا كان كلا البتتين 1
	OR	يضبط كل بت على 1 إذا كان أحد البتتين هو 1
^	XOR	يضبط كل بت على 1 إذا كان واحد فقط من البتتين هو 1
~	NOT	يعكس كل البتات
<<	Zero fill left shift	ينتقل إلى اليسار عن طريق دفع الأصفار من اليمين وترك البتات الموجودة في أقصى اليسار تسقط
>>	Signed right shift	ينتقل إلى اليمين عن طريق دفع نسخ من أقصى اليسار إلى الداخل من اليسار، ويترك البتات الموجودة في أقصى اليمين تسقط
>>>	Zero fill right shift	ينتقل إلى اليمين عن طريق دفع الأصفار من اليسار، ويترك البتات الموجودة في أقصى اليمين تسقط

أمثلة

المعامل	النتيجة	مثال	النتيجة
5 & 1	1	0101 & 0001	0001
5 1	5	0101 0001	0101
~ 5	10	~0101	1010
5 << 1	10	0101 << 1	1010
5 ^ 1	4	0101 ^ 0001	0100
5 >> 1	2	0101 >> 1	0010
5 >>> 1	2	0101 >>> 1	0010

Bitwise XOR

على زوج من البتات، فإنه يُرجع 1 إذا كانت البتات مختلفة XOR عندما يتم تنفيذ

4: بت (↓↓ مثال ✓)

عملية	نتيجة
1111 ^ 0000	1111
1111 ^ 0001	1110
1111 ^ 0010	1101
1111 ^ 0100	1011

البتتين 1:

Bitwise AND (&)

يُرجع 1 فقط إذا كانت كلا Bitwise AND

عدد عشري	الثنائية
5	000000000000000000000000101
1	000000000000000000000000001
5 و 1	000000000000000000000000001 (1)

(↓↓ مثال ✓)

let x = 5 & 1;

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Bitwise أو (|)

يُرجع 1 إذا كانت إحدى البتات هي Bitwise OR 1

عدد عشري	الثنائية
5	000000000000000000000000101
1	000000000000000000000000001
5 1	000000000000000000000000101 (5)

(↓↓ مثال ✓)

let x = 5 | 1;

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Bitwise XOR (^)

يلرجع 1 إذا كانت البتات مختلفة Bitwise XOR يقوم

عدد عشري	الثنائية
5	000000000000000000000000101
1	000000000000000000000000001
$5 \wedge 1$	000000000000000000000000100 (4)

(مثال ✓ ↓↓)

`let x = 5 ^ 1;`

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

Bitwise NOT (~)

عدد	الثنائية
5	000000000000000000000000101
~ 5	1111111111111111111111111010 (-6)

(مثال ✓ ↓↓)

`let x = ~5;`

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

كيف إزاحة البت لليسار بمقدار (<<)

هذا هو التحول الأيسر ملء الصفر. يتم دفع بت واحد أو أكثر من الصفر من اليمين، وتسقط البتات الموجودة في أقصى اليسار

عدد	الثنائية
5	000000000000000000000000101
$5 \ll 1$	000000000000000000000001010 (10)

(مثال ✓ ↓↓)

`let x = 5 << 1;`

علامة الإزاحة إلى اليمين (>>)

هذه علامة تحافظ على التحول الصحيح. يتم دفع نسخ من البت الموجود في أقصى اليسار إلى الداخل من اليسار، وتسقط البتات الموجودة في أقصى اليمين:

عدد عشري	الثنائية
-5	1111111111111111111111111111011
-5 >> 1	111111111111111111111111111101 (-3)

(مثال ✓ ↓↓)

```
let x = -5 >> 1;
```

علامة التحول الأيمن (>>>)

هذا هو التحول الأيمن ملء الصفر. يتم دفع بت واحد أو أكثر من الصفر من اليسار، وتسقط البتات الموجودة في أقصى اليمين:

عدد عشري	الثنائية
5	0000000000000000000000000000101
5 >>> 1	000000000000000000000000000010 (2)

(مثال ✓ ↓↓)

```
let x = 5 >>> 1;
```

ترجمة الأرقام إلى لغة البيناري الثانية

من السهل فهم الأرقام الثنائية التي تحتوي على مجموعة بت واحدة فقط:

الترجمة الثنائي	القيمة
00000000000000000000000000000001	1
00000000000000000000000000000010	2
00000000000000000000000000000100	4

البرامترات العادية

البرامتر العادي عبارة عن نص من الأحرف التي تشكل نمط بحث يمكن استخدام نمط البحث في عمليات البحث عن النص واستبدال النص.

ما هو البرامتر العادي؟

البرامتر العادي عبارة عن نص من الأحرف التي تشكل نمط البحث. عندما تبحث عن كلمات في نص ما، يمكنك استخدام نمط البحث هذا لوصف ما تبحث عنه. يمكن أن يكون البرامتر العادي حرفاً واحداً أو نمطاً أكثر تعقيداً. يمكن استخدام البرامترات العادية لإجراء جميع أنواع عمليات البحث عن النص واستبدال النص.

قوانين البحث

(مثال ✓) ↓↓

/Abu Habib alhosiny /i;

شاهد ال (✓ مثال ↓↓)

/Abu Habib alhosiny /i هو تعبير عادي

هو نمط (يستخدم في البحث)

هو مُعدّل (يعدل البحث ليكون غير حساس لحالة الأحرف) **i**

استخدام دوال النص

search() و **replace()**: في جافاسكربت، غالباً ما تُستخدم البرامترات العادية مع طريقتي النص

تعبيراً للبحث عن تطابق، وتقوم بإرجاع موضع التطابق **search()** تستخدم الوظيفة

إرجاع نص معدلة حيث يتم استبدال الخاصية **replace()** تقوم الوظيفة

طرق البحث في النصوص

:عن نص لقيمة محددة وترجع موضع المطابقة (`search()`) تبحث الوظيفة

(مثال $\Downarrow\Downarrow$ ✓)

: في نص "Abu Habib alhosiny" استخدم نص لإجراء بحث عن

```
let abib = "Visit Abu Habib alhosiny !";  
let n = abib.search("Abu Habib alhosiny ");
```

:ستكون n النتيجة في

6

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

أحد طرق البحث الغير حساس لحالة الأحرف

(مثال ✓ ↓↓)

: في نص "Abu Habib alhosiny" استخدم تعبيرًا عاديًا لإجراء بحث غير حساس لحالة الأحرف عن

```
let abib = "Visit Abu Habib alhosiny ";  
let n = abib.search(/Abu Habib alhosiny /i);
```

:ستكون n النتيجة في

6

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

كيف البحث و الاستبدال

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

```
let abib = "Visit Hamza!";  
let result = abib.replace("Hamza", "Abu Habib alhosiny ");
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

طرق اخرى لاستبدال الغير حساس للحالة

(مثال ✓ ↓↓)

: في نص Abu Habib alhosiny بـ Hamza استخدم تعبيرًا عاديًا غير حساس لحالة الأحرف لاستبدال


```
let abib = "Visit Hamza!";
```

```
let result = abib.replace(/Hamza/i, "Abu Habib alhosiny ");
```

النتيجة في المكون ستكون

```
//Visit Abu Habib alhosiny !
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

هل لاحظت؟

يمكن استخدام وسيطات اليرامتر العادي (بدلاً من وسيطات النص) في الطرق المذكورة أعلاه. يمكن أن تجعل اليرامترات العادية بحثك أكثر قوة (غير حساس لحالة الأحرف على سبيل المثال) (مثال ↴).

معدّلات اليرامتر العادي

يمكن استخدام المعدلات لإجراء المزيد من عمليات البحث العامة غير الحساسة لحالة الأحرف:

Modifier	وصف
i	إجراء مطابقة غير حساسة لحالة الأحرف
g	إجراء مطابقة عامة (البحث عن كافة التتابعات بدلاً من التوقف بعد الإجراء الأولي)
m	إجراء مطابقة متعددة الأسطر

خصائص اليرامتر العادية

تستخدم الأقواس للعثور على مجموعة من الأحرف:

Expression	وصف
[abc]	ابحث عن أي من الأحرف الموجودة بين قوسين
[0-9]	ابحث عن أي من الأرقام الموجودة بين القوسين
(x y)	ابحث عن أي من البدائل مفصولة بـ

الأحرف الأولية هي أحرف ذات معنى خاص:

Metacharacter	وصف
<code>\d</code>	ابحث عن رقم
<code>\s</code>	ابحث عن حرف مسافة فارغة
<code>\b</code>	أو في نهاية كلمة، <code>\bWORD</code> : ابحث عن تطابق في بداية كلمة مثل هذه <code>WORD\b</code> : مثل هذه
<code>\uxxxx</code>	المحدد بالرقم السداسي العشري Unicode ابحث عن حرف <code>xxxx</code>

تحدد الكميات الكميات:

Quantifier	وصف
<code>n+</code>	واحد على الأقل <code>n</code> يطابق أي نص تحتوي على
<code>n*</code>	<code>n</code> يطابق أي نص تحتوي على صفر أو أكثر من تكرارات
<code>n?</code>	<code>n</code> يطابق أي نص تحتوي على صفر أو تكرار واحد لـ

RegExp باستخدام كائن

هو كائن تعبير عادي له خصائص ودوال محددة مسبقًا `RegExp` في جافاسكربت، كائن سنتحدث عنه ان شاء الله.

أساليب الاختبار

`RegExp` هي وظيفة تعبير `test()` الوظيفة

يبحث في نص عن نمط، ويعيد صواب أو خطأ، اعتمادًا على النتيجة

"e": يبحث ال (✓ مثال ↓↓) التالي عن نص للحرف

(✓ مثال ↓↓)

```
const pattern = /e/;
pattern.test("Abu Habeb Al Husene *_* ___");
```

في النص، فإن مخرجات الكود أعلاه ستكون "e" بما أن هناك حرف

true

ليس عليك وضع البرامتر العادي في متغير أولاً. يمكن اختصار السطرين أعلاه إلى سطر واحد

```
/e/.test("Abu Habib Al Husini *_* ___");
```

تابع طرق البحث

RegExp هي وظيفة تعبير (`exec()`) الوظيفة

• يبحث في نص عن نمط محدد، ويعيد النص الذي تم العثور عليه ككائن

• إذا لم يتم العثور على أي تطابق، فسيتم إرجاع كائن فارغ (فارغ)

• "e" يبحث ال (✓ مثال ↓↓) التالي عن نص للحرف

(✓ مثال ↓↓)

```
/e/.exec("Abu Habeb Al Husene *_* ___");
```

أسبقية المشغل

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

•+- () الضرب (*) والقسمة (/) لهما أسبقية / أعلى من الجمع (+) والطرح

كما هو الحال في الرياضيات التقليدية، يتم الضرب أولاً

let x = 100 + 50 * 3;

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

عند استخدام الأقواس، يتم حساب العمليات داخل الأقواس أولاً

let x = (100 + 50) * 3;

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

يتم حساب العمليات التي لها نفس الأولوية (مثل * و /) من اليسار إلى اليمين

let x = 100 / 50 * 3;

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

التعامل مع الأخطاء في جافاسكربت

التقاط الخطأ دون مشاكل

جملة تراهي التي سنستخدمها الآن للامساك بالأخطاء وكشفها وإرسالها فهي تحاول تنفيذ الكود فان نفذته خير وبركة وان لم تنفذ
تمسك الخطأ وترسله

كتلة ال اكواد للتشغيل (للمحاولة) **try** يحدد الكلمة

كتلة ال اكواد للتعامل مع أي خطأ **catch** يحدد الكلمة

كتلة ال اكواد للتشغيل بغض النظر عن النتيجة **finally** يحدد الكلمة

خطأ مخصص **throw** يحدد الكلمة

عند حدوث الخطأ

عند تنفيذ كود جافاسكربت، يمكن أن تحدث أخطاء مختلفة.

يمكن أن تكون الأخطاء عبارة عن أخطاء ترميزية يقوم بها المبرمج، وأخطاء ناتجة عن إدخال خاطئ، وأشياء أخرى غير متوقعة.

(↓↓↓ مثال ✓)

: لإنتاج خطأ عمدًا لا يوجد شيء اسمه "addlert" الاسم الصحيح "alert" في هذا ال(✓ مثال ↓↓↓)، أخطأنا في كتابة

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

```
<script>
try {
  addlert("Welcome Habib*_*!");
}
catch(err) {
  document.getElementById("Habib").innerHTML = err.message;
}
</script>
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

ت، وتنفذ رمز الالتقاط للتعامل معه Addler نكتشف جافاسكربت خطأ

عبيلة وأديلة 🤪

لا تستغرب من هذا العنوان لأنه يصف جملة ترى إذا كتبت كل ال اكواد بداخلها ستجمع لك جميع الاخطاء وترسلها اليك . لتعرفها

تحديد كتلة من ال اكواد ليتم اختبارها بحثًا عن الأخطاء أثناء تنفيذها **try** يتيح لك الكلمة

.تحديد كتلة من ال اكواد التي سيتم تنفيذها، في حالة حدوث خطأ في كتلة المحاولة **catch** يتيح لك الكلمة

:في أزواج **catch** وتأتي **try** عبارات جافاسكربت

```
try {
  Block of code to try
}
catch(err) {
```

Block of code to handle errors

```
}
```

ماذا يحدث عند الإخطاء

عند حدوث خطأ، ستتوقف جافاسكربت عادةً وتنتج رسالة خطأ

. المصطلح الفني لذلك هو: سوف يقوم جافاسكربت بطرح استثناء (رمي خطأ)

. بخاصيتين : الاسم والرسالة ونوع الخطأ Error ستقوم جافاسكربت فعليًا بإنشاء كائن

التحكم في الإخطاء

بإنشاء خطأ مخصص **throw** يسمح لك الكلمة

. من الناحية الفنية يمكنك طرح استثناء (رمي خطأ)

Object أو **a** أو **Boolean** أو **String** يمكن أن يكون الاستثناء جافاسكربت

```
throw "Too big"; // throw a abib  
throw 500; // throw a number
```

فيمكنك التحكم في تدفق البرنامج وإنشاء رسائل خطأ مخصصة، **try catch** مع **throw** إذا كنت تستخدم

للتحقق من صحة الإدخال

يفحص هذا ال (مثال ↓↓) المدخلات. إذا كانت القيمة خاطئة، فسيتم طرح استثناء (يخطئ)

:بواسطة عبارة الالتقاط ويتم عرض رسالة خطأ مخصصة (**err**) يتم اكتشاف الاستثناء

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

```
<p>Please input a number between 5 and 10:</p>
```

```
<input id="Habib" type="abib">  
<button type="button" onclick="Husini()">Test Input</button>  
<p id="habib3"></p>
```



```
<script>
function Husini() {
  const message = document.getElementById("habib3");
  message.innerHTML = "";
  let x = document.getElementById("Habib").value;
  try {
    if(x.trim() == "") throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5) throw "too low";
    if(x > 10) throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
}
</script>

</body>
</html>
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

التحقق من صحة HTML

الكود أعلاه هو مجرد (مثال ⚡)

المضمن، وذلك باستخدام قواعد HTML غالباً ما تستخدم المتصفحات الحديثة مزيجاً من جافاسكربت والتحقق من صحة HTML: التحقق المحددة مسبقاً والمحددة في سمات

```
<input id="Habib" type="number" min="5" max="10" step="1">
```

يمكنك قراءة المزيد حول التحقق من صحة النماذج في فصل لاحق من هذا الكتاب.

البرامتر الأخير

تنفيذ ال اكواد ، بعد المحاولة والالتقاط، بغض النظر عن النتيجة **finally** تتيح لك العبارة

بناء الجملة

```
try {  
  Block of code to try  
}  
catch(err) {  
  Block of code to handle errors  
}  
finally {  
  Block of code to be executed regardless of the try / catch result  
}
```

(مثال ↓↓↓)

```
function Husini() {  
  const message = document.getElementById("habib3");  
  message.innerHTML = "";  
  let x = document.getElementById("Habib").value;  
  try {  
    if(x.trim() == "") throw "is empty";  
    if(isNaN(x)) throw "is not a number";  
    x = Number(x);  
    if(x > 10) throw "is too high";  
    if(x < 5) throw "is too low";  
  }  
  catch(err) {  
    message.innerHTML = "Error: " + err + ".";  
  }  
  finally {  
    document.getElementById("Habib").value = "";  
  }  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

كائن الخطأ

يحتوي جافاسكربت على كائن خطأ مضمن يوفر معلومات الخطأ عند حدوث خطأ

يوفر كائن الخطأ خاصيتين مفيدتين: الاسم والرسالة

قيم اسم الخطأ

يمكن إرجاع ست قيم مختلفة بواسطة خاصية اسم الخطأ:
يتم وصف القيم الست المختلفة أدناه

خطأ في التقييم

`eval()` إلى خطأ في وظيفة `EvalError` يشير

بدلاً من ذلك `SyntaxError` استخدم `EvalError` الإصدارات الأحدث من جافاسكربت لا ترمي

خطأ في النطاق

إذا كنت تستخدم رقمًا يقع خارج نطاق القيم القانونية `RangeError` يتم طرح
على سبيل المثال (مثال): لا يمكنك تعيين عدد الأرقام المهمة لرقم ما على 500

(مثال)

```
let num = 1;
try {
  num.toPrecision(500); // A number cannot have 500 significant digits
}
catch(err) {
  document.getElementById("Habib").innerHTML = err.Name;
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

خطأ مرجعي

إذا كنت تستخدم (مرجعًا) متغيرًا لم يتم الإعلان عنه `ReferenceError` يتم طرح

(↓↓↓ مثال ✓)

```
let x = 5;
try {
  x = y + 1; // y cannot be used (referenced)
}
catch(err) {
  document.getElementById("Habib").innerHTML = err.Name;
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

خطأ في بناء الجملة

إذا حاولت تقييم ال اكواد التي تحتوي على خطأ في بناء الجملة **SyntaxError** A يتم طرح

(↓↓↓ مثال ✓)

```
try {
  eval("alert('Easy-to)"); // Missing ' will produce an error
}
catch(err) {
  document.getElementById("Habib").innerHTML = err.Name;
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

كيف معرفة نوع الخطأ

إذا كنت تستخدم قيمة خارج نطاق الأنواع المتوقعة **TypeError** A يتم طرح

(↓↓↓ مثال ✓)

```
let num = 1;
try {
  num.toUpperCase(); // You cannot convert a number to upper case
}
catch(err) {
```



```
document.getElementById("Habib").innerHTML = err.Name;
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

كيف معرفة اخطا الروابط

URI: إذا كنت تستخدم أحرفاً غير قانونية في دالة **URIError** A يتم طرح

(مثال ✓ ↓↓↓)

```
try {
  decodeURI("%%%"); // You cannot URI decode percent signs
}
catch(err) {
  document.getElementById("Habib").innerHTML = err.Name;
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

نطاق الكتلة

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

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

.توفر هاتان الكلمتان الرئيسيتان نطاق الكتلة في جافاسكربت

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

(مثال ✓ ↓↓↓)

```
{
  let x = 2;
```

```
}  
// x can NOT be used here
```

الكلمة المحجوزة أن تحتوي على نطاق حظر **var** لا يمكن للمتغيرات المعلنه باستخدام
يمكن الوصول إلى المتغيرات المعلنه داخل الكتلة { } من خارج الكتلة

(مثال ✓ ↓↓)

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

النطاق المحلي

المتغيرات المعلنه داخل وظيفة جافاسكربت، تصبح محلية للوظيفة

(مثال ✓ ↓↓)

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

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

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

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

: المتغيرات المحلية لها نطاق الوظيفة

ولا يمكن الوصول إليها إلا من داخل الوظيفة

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

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

نطاق الوظيفة

لدى جافاسكربت نطاق وظيفي: كل وظيفة تنشئ نطاقاً جديداً.
لا يمكن الوصول إلى المتغيرات المحددة داخل الوظيفة (مرئية) من خارج الوظيفة.
متشابهة تماماً عند الإعلان عنها داخل دالة **const** تكون **let**، **var** المتغيرات المعلنة به.
لديهم جميعاً نطاق الوظيفة:

```
function Husini() {  
  var hAbiB4 = "Hamza"; // Function Scope  
}
```

```
function Husini() {  
  let hAbiB4 = "Hamza"; // Function Scope  
}
```

```
function Husini() {  
  const hAbiB4 = "Hamza"; // Function Scope  
}
```

متغيرات العامة

GLOBAL المتغير المعلن خارج الدالة يصبح

(مثال ✓ ↓↓)

```
let hAbiB4 = "Hamza";  
// code here can use hAbiB4
```

```
function Husini() {  
  // code here can also use hAbiB4  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

: المتغير العام له نطاق عام

يمكن لجميع الكود والوظائف الموجودة على صفحة الويب الوصول إليها

النطاق العام

المتغيرات المعلنة عاميًا (خارج أي وظيفة) لها نطاق عام

يمكن الوصول إلى المتغيرات العامة من أي مكان في برنامج جافاسكربت

متشابهة تمامًا عندما يتم الإعلان عنها خارج الكتلة **const** تكون **var**، **let** المتغيرات المعلنة بـ

: لديهم جميعا نطاق عام

```
| var x = 2; // Global scope
```

```
| let x = 2; // Global scope
```

```
| const x = 2; // Global scope
```

متغيرات

في جافاسكربت، الكائنات والوظائف هي أيضًا متغيرات

. يحدد النطاق إمكانية الوصول إلى المتغيرات والكائنات والوظائف من أجزاء مختلفة من ال اكواد

العامة التلقائية

. إذا قمت بتعيين قيمة لمتغير لم يتم التصريح عنه، فسوف يصبح تلقائيًا متغيرًا عاميًا

حتى لو تم تعيين القيمة داخل دالة، **hAbiB4** سيعلن (مثال ↴) ال اكواد هذا عن متغير عام

(مثال ↴)

```
| Husini();
```

```
| // code here can use hAbiB4
```

```
| function Husini() {
```



```
hAbiB2= "Hamza";  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

الوضع الصارم

"تدعم جميع المتصفحات الحديثة تشغيل جافاسكربت في "الوضع الصارم

في "الوضع الصارم"، لا تصبح المتغيرات غير المعلنة عمومية تلقائيًا

المتغيرات العامة في HTML

مع جافاسكربت، النطاق العام هو بيئة جافاسكربت

النطاق العام هو كائن النافذة HTML، في

المحجوزة تنتمي إلى كائن النافذة **var** المتغيرات العامة المحددة بالكلمة

(↓↓↓ مثال ✓)

```
var hAbiB4 = "Hamza";  
// code here can use window.hAbiB4
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

المحجوزة لا تنتمي إلى كائن النافذة **let** المتغيرات العامة المحددة بالكلمة

(↓↓↓ مثال ✓)

```
let hAbiB4 = "Hamza";  
// code here can not use window.hAbiB4
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

تحذير

لا تقم بإنشاء متغيرات عمومية إلا إذا كنت تقصد ذلك.

يمكن للمتغيرات العامة (أو الوظائف) أن تحل محل متغيرات النافذة (أو الوظائف) يمكن لأي وظيفة، بما في ذلك كائن النافذة، استبدال المتغيرات والوظائف العامة.

عمر المتغيرات

يبدأ عمر متغير جافاسكربت عند الإعلان عنه.

يتم حذف متغيرات الوظيفة (المحلية) عند اكتمال الوظيفة.

في متصفح الويب، يتم حذف المتغيرات العامة عند إغلاق نافذة (أو علامة التبويب) المتصفح.

البرمترات في الوظائف

تعمل وسيطات الدالة (البرمترات) كمتغيرات محلية داخل الوظائف.

إعلان عن متغير

في جافاسكربت، يمكن الإعلان عن المتغير بعد استخدامه.

بعبارة أخرى؛ يمكن استخدام المتغير قبل الإعلان عنه.

: يعطي ال (✓ مثال ↓↓) 1 نفس نتيجة ال (✓ مثال ↓↓) 2

1 (✓ مثال ↓↓)

```
| x = 5; // Assign 5 to x
```

```
| elem = document.getElementById("Habib"); // Find an element
```

```
| elem.innerHTML = x; // Display x in the element
```

```
| var x; // Declare x
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

2 (↓↓↓ مثال ✓)

```
var x; // Declare x
x = 5; // Assign 5 to x

elem = document.getElementById("Habib"); // Find an element
elem.innerHTML = x; // Display x in the element
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

"لفهم هذا، عليك أن تفهم مصطلح "الرفع

الرفع هو السلوك الافتراضي لجافاسكربت المتمثل في نقل كافة الإعلانات إلى أعلى النطاق الحالي (إلى أعلى الكود الحالي أو الوظيفة الحالية).

أفضليات `const` و `let`

• إلى أعلى الكتلة، ولكن لم تتم تهيئتها `const` ورفعها `let` المتغيرات التي تم تعريفها.
المعنى: كتلة ال اكواد على علم بالمتغير، لكن لا يمكن استخدامها حتى يتم الإعلان عنها.
`ReferenceError` سيؤدي استخدام متغير قبل الإعلان عنه إلى `let`.
المتغير موجود في "منطقة متروكة مؤقتة" من بداية الكتلة حتى يتم الإعلان عنه

(↓↓↓ مثال ✓)

`ReferenceError`: سيؤدي هذا إلى

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

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

متغير قبل الإعلان عنه خطأً في بناء الجملة، لذا لن يتم تشغيل ال اكواد ببساطة `const` يعد استخدام

(↓↓↓ مثال ✓)

لن يتم تشغيل هذا الرمز

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

```
const hAbiB4;
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

لا يتم رفع عمليات التهيئة

تقوم جافاسكريبت برفع الإعلانات فقط، وليس عمليات التهيئة

: ال (✓ مثال ↓↓) 1 لا يعطي نفس نتيجة ال (✓ مثال ↓↓) 2

1 (✓ مثال ↓↓)

```
var x = 5; // Initialize x
```

```
var y = 7; // Initialize y
```

```
elem = document.getElementById("Habib"); // Find an element
```

```
elem.innerHTML = x + " " + y; // Display x and y
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

2 (✓ مثال ↓↓)

```
var x = 5; // Initialize x
```

```
elem = document.getElementById("Habib"); // Find an element
```

```
elem.innerHTML = x + " " + y; // Display x and y
```

```
var y = 7; // Initialize y
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

غير محدد في ال (✓ مثال ↓↓) الأخير؟ y هل يعقل أن

فقط، وليس التهيئة ($7=$) إلى الأعلى ($\text{var } y$) وذلك لأنه يتم رفع الإعلان

غير محددة y قبل استخدامه، ولكن نظرًا لعدم رفع عمليات التهيئة، فإن قيمة y بسبب الرفع، تم الإعلان عن

: ال (✓ مثال ↓↓) 2 هو نفس الكتابة

(مثال ↓↓↓)

```
var x = 5; // Initialize x
var y; // Declare y

elem = document.getElementById("Habib"); // Find an element
elem.innerHTML = x + " " + y; // Display x and y

y = 7; // Assign 7 to y
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

أعلن عن المتغيرات في الأعلى

يعد الرفع (بالنسبة للعديد من المطورين) سلوكًا غير معروف أو يتم التغاضي عنه في جافاسكريبت. إذا لم يفهم المطور الرفع، فقد تحتوي البرامج على أخطاء (أخطاء) لتجنب الأخطاء، قم دائمًا بإعلان كافة المتغيرات في بداية كل نطاق. نظرًا لأن هذه هي الوظيفة التي تفسر بها جافاسكريبت ال اكواد ، فهي دائمًا قاعدة جيدة. لا تسمح جافاسكريبت في الوضع الصارم باستخدام المتغيرات إذا لم يتم الإعلان عنها.

استخدم الكلمة Strict

"يحدد أنه يجب تنفيذ تعليمات جافاسكريبت البرمجية في "الوضع الصارم"; "use strict".

"توجيه" الاستخدام الصارم.

ECMAScript كان التوجيه جديدًا في الإصدار 5 من "use strict".

إنه ليس عبثًا، ولكنه تعبير حرفي، تم تجاهله في الإصدارات السابقة من جافاسكريبت. "هو الإشارة إلى أنه يجب تنفيذ ال اكواد في "الوضع الصارم" "use strict" والغرض من ذلك مع الوضع الصارم، لا يمكنك، على سبيل ال (مثال ↓↓↓)، استخدام متغيرات غير معلنة.

تدعم كافة المتصفحات الحديثة "الاستخدام الصارم" باستثناء 9 والإصدارات الأقدم.
تحدد الأرقام الموجودة في الجدول إصدار المتصفح الأول الذي يدعم التوجيه بشكل كامل.

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

بإلقاء خطأ حتى لو لم يفهمه IE 9 هي مجرد نص ، لذلك لن يقوم "use strict".

إعلان الوضع الصارم

يتم الإعلان عن الوضع الصارم بإضافة "استخدام صارم"؛ هذه معنى الكلمة إلى بداية الكود أو الوظيفة:
تم الإعلان عنه في بداية الكود، وله نطاق عام (سيتم تنفيذ جميع ال اكواد الموجودة في الكود في الوضع الصارم)

(↓↓↓ مثال ✓)

```
"use strict";  
x = 3.14; // This will cause an error because x is not declared
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

(↓↓↓ مثال ✓)

```
"use strict";  
Husini();  
  
function Husini() {  
y = 3.14; // This will also cause an error because y is not declared  
}
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

تم الإعلان عنها داخل دالة، ولها نطاق محلي (فقط الكود الموجود داخل الوظيفة هو في الوضع الصارم)

```
x = 3.14; // This will not cause an error.  
Husini();
```

```
function Husini() {  
"use strict";
```



```
y = 3.14; // This will cause an error
```

```
}
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

تابع استخدام الوضع الصارم

تم تصميم بناء الجملة، للإعلان عن الوضع الصارم، ليكون متوافقاً مع الإصدارات الأقدم من جافاسكربت لانها كلمة مكتوبة مثل هذه الجملة في ليس له أي آثار . ("Abu Bakr Al-Siddiq") تجميع حرفي أو نص حرفية . ببساطة يجمع إلى متغير غير موجود و لا يحدث خطأ في المتصفحات القديمة التي لا تعرف شى عن الوضع الصارم . لا يهم إلا المتصفحات الجديده الذين معناه: "use strict" لذلك

لماذا الوضع الصارم؟

"الوضع الصارم يجعل من السهل كتابة جافاسكربت "الآمنة".
يغير الوضع الصارم "بناء الجملة السيئ" المقبول مسبقاً إلى أخطاء حقيقية على سبيل ال (مثال خطأ)، في جافاسكربت العادية، يؤدي الخطأ في كتابة اسم المتغير إلى إنشاء متغير عام جديد. في الوضع الصارم، سيؤدي هذا إلى حدوث خطأ، مما يجعل من المستحيل إنشاء متغير عام عن طريق الخطأ .
في جافاسكربت العادية، لن يتلقى المطور أي تعليقات خطأ في تعيين قيم لخصائص غير قابلة للكتابة فقط، أو خاصية غير موجودة، أو متغير غير getter في الوضع الصارم، أي إسناد إلى خاصية غير قابلة للكتابة، أو خاصية موجودة، أو كائن غير موجود، سيؤدي إلى حدوث خطأ

الغير مسموح به في الوضع الصارم

:لا يجوز استخدام المتغير دون الإعلان عنه

```
"use strict";
```

```
x = 3.14; // This will cause an error
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

.الكائنات هي متغيرات أيضا

:لا يجوز استخدام كائن دون الإعلان عنه

```
"use strict";  
x = {p1:10, p2:20}; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

غير مسموح بحذف متغير (أو كائن)

```
"use strict";  
let x = 3.14;  
delete x; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

غير مسموح بحذف وظيفة

```
"use strict";  
function x(p1, p2) {};  
delete x; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

لا يُسمح بتكرار اسم المعلمة

```
"use strict";  
function x(p1, p1) {}; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

غير مسموح بالأحرف الرقمية الثماني

```
"use strict";  
let x = 010; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

غير مسموح بأحرف الاستثناء الثمانية


```
"use strict";  
let x = "\010"; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

:الكتابة إلى خاصية للقراءة فقط غير مسموح بها

```
"use strict";  
const obj = {};  
Object.defineProperty(obj, "x", {value:0, writable:false});  
  
obj.x = 3.14; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

:لا يُسمح بالكتابة إلى خاصية الحصول عليها فقط

```
"use strict";  
const obj = {get x() {return 0}};  
  
obj.x = 3.14; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

:لا يُسمح بحذف خاصية غير قابلة للحذف

```
"use strict";  
delete Object.prototype; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

:كمتغير **eval** لا يمكن استخدام الكلمة

```
"use strict";  
let eval = 3.14; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

كمتغير **arguments** لا يمكن استخدام الكلمة

```
"use strict";  
let arguments = 3.14; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

لا يجوز الكلمة **with**:

```
"use strict";  
with (Math){x = cos(2)}; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

لا يسمح بإنشاء متغيرات في النطاق الذي تم استدعاؤه منه **eval()**، لأسباب أمنية

في الوضع الصارم، لا يمكن استخدام المتغير قبل الإعلان عنه

```
"use strict";  
eval ("x = 2");  
alert (x); // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

var: الإعلان عن متغير باستخدام الكلمة المحجوزة **eval()** في الوضع الصارم، لا يمكن لـ

```
"use strict";  
eval ("var x = 2");  
alert (x); // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

Let: الإعلان عن متغير باستخدام الكلمة المحجوزة **eval()** لا يمكن لـ


```
eval ("let x = 2");  
alert (x); // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

المحجوزة في الوظائف بشكل مختلف في الوضع الصارم **this** تتصرف الكلمة
المحجوزة إلى الكائن الذي يسمى الوظيفة **this** تشير الكلمة
وستعيد الوظائف في الوضع العادي الكائن العام **undefined** إذا لم يتم تحديد الكائن، فستعود الوظائف في الوضع الصارم
(النافذة):

```
"use strict";  
function Husini() {  
  alert(this); // will alert "undefined"  
}  
Husini();
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

تابع الوضع الصارم

لا يمكن استخدام الكلمات المحجوزة لإصدارات جافاسكريبت المستقبلية كأسماء متغيرات في الوضع الصارم

```
"use strict";  
let public = 1500; // This will cause an error
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

!احترس

يتم التعرف على التوجيه "استخدام صارم" فقط في بداية البرنامج النصي أو الوظيفة

هذه الكلمة المحجوزة

(مثال ↓↓) ✓

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

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

الكلمة زيس

. تشير الكلمة المحجوزة إلى كائن **this**، في جافاسكربت
. استدعائه (استخدامه أو استدعائه) **this** يعتمد الكائن على كيفية
:المحجوزة إلى كائنات مختلفة اعتمادًا على كيفية استخدامها **this** تشير الكلمة

ملحوظة

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

الإشارة من داخل الدالة

. فإنه يشير إلى الكائن **this**، عند استخدامه في وظيفة كائن
. يشير إلى كائن العنصر **this**، في ال (مثال ↓↓) الموجود أعلى هذه الصفحة
. لأن وظيفة الاسم الكامل هي وظيفة لكائن العنصر


```
fullName : function() {  
  return this.firstName + " " + this.lastName;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

تابع كلمة زيس

• يشير إلى الكائن العام **this**، عند استخدامه بمفرده.
• يعمل في النطاق العام **this** لأنه
[object Window]: الكائن العام في نافذة المتصفح هو

(مثال ✓ ↓↓)

```
let x = this;
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

: يشير أيضًا إلى الكائن العام **this**، في الوضع الصارم، عند استخدامه بمفرده

(مثال ✓ ↓↓)

```
"use strict";  
let x = this;
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

كلمة هذا في الوضع العادي

this في إحدى الوظائف، يكون الكائن العام هو الرابط الافتراضي لـ
[object Window]: الكائن العام في نافذة المتصفح هو

(مثال ✓ ↓↓)

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

كلمة هذا في الوضع الصارم

لا يسمح الوضع الصارم لجافاسكريبت بالربط الافتراضي `undefine` يكون `this`، لذا، عند استخدامه في دالة، في الوضع الصارم

(مثال ↓↓) ✓

```
"use strict";  
function Husini() {  
  return this;  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

كلمة هذا في معالجات الأحداث

الذي استقبل الحدث HTML يشير إلى عنصر `this`، HTML في معالجات أحداث

(مثال ↓↓) ✓

```
<button onclick="this.style.display='none'">  
  Click to Remove Me!  
</button>
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

ربط وظيفة الكائن

هو العنصر الكائن `this`، في هذه الأمثلة

(↓ ↓ مثال ✓)

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

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

(↓ ↓ مثال ✓)

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

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

هو خاصية الاسم الأول لهذا (كائن العنصر) `this.firstName`، على سبيل ال (✓ مثال ↓ ↓)

ربط الوظيفة الصريحة

هي دوال جافاسكريبت محددة مسبقاً `apply()` والدوال `call()` الدوال. يمكن استخدامهما لاستدعاء أسلوب كائن مع كائن آخر كوسيلة.

أنظر أيضا:

حتى لو كان `hAbiB2` كوسيلة، ويشير هذا إلى `hAbiB2` مع `hAbiB1.fullName` يستدعي ال (✓ مثال ↓ ↓) أدناه `hAbiB1` تابعًا لـ `fullName`:

(مثال ↓↓↓)

```
const hAbiB1 = {
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}
```

```
const hAbiB2 = {
  firstName:"Habib",
  lastName: "Al Husini 🇸🇩🇸🇩🇸🇩",
}
```

```
hAbiB1.fullName.call(hAbiB2);
```

```
// Return "Habib Al Husini 🇸🇩🇸🇩🇸🇩":
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

تابع وظائف الكائن

الوظيفة، يمكن للكائن استعارة وظيفة من كائن آخر (`bind()`) باستخدام

يقوم هذا ال (مثال ↓↓↓) بإنشاء كائنين (عنصر وعضو)

يستعير كائن العضو وظيفة 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 الثاني ابو حبيب الحسيني

الوظيفة المختصرة

ES6. تم تقديم وظائف مختصرة في

تسمح لنا الدوال مختصرة بكتابة صيغة دالة أقصر

```
let Husini = (a, b) => a * b;
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

: تابع الوظائف المختصرة

```
Easy-to = function() {  
  return "Abu Habib Al Husini *_*!";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

:وظيفة مختصرة اخرى

```
Easy-to = () => {  
  return "Abu Habib Al Husini *_*!";  
}
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

يصبح أقصر! إذا كانت الدالة تحتوي على عبلة واحدة فقط، وكانت العبلة تُرجع قيمة، فيمكنك إزالة **return** الأقواس والكلمة المحجوزة

:الدوال المختصرة تُرجع القيمة افتراضياً

| **Easy-to = () => "Abu Habib Al Husini *_*!";**

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

ملاحظة: يعمل هذا فقط إذا كانت الدالة تحتوي على عبلة واحدة فقط

:إذا كان لديك معلمات، يمكنك تمريرها داخل الأقواس

:الوظيفة المختصرة مع البرمترات

| **Easy-to = (val) => "Easy-to " + val;**

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

:في الواقع، إذا كان لديك معلمة واحدة فقط، فيمكنك تخلي الأقواس أيضاً

:دالة المختصرة بدون الأقواس

| **Easy-to = val => "Easy-to " + val;**

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

this في الوظائف المختصرة

يختلف التعامل أيضاً في وظائف المختصرة مقارنة بالوظائف العادية **this**

this باختصار، مع وظائف المختصرة لا يوجد ربط لـ

تمثل الكلمة المحجوزة الكائن الذي يسمى الوظيفة، والذي يمكن أن يكون النافذة أو المستند أو الزر **this**، في الوظائف العادية، أو أي شيء آخر.

المحجوزة دائماً الكائن الذي يحدد وظيفة المختصرة **this** مع وظائف الأمختصرة، تمثل الكلمة

دعونا نلقي نظرة على **المثالين** لفهم الفرق

كلا **المثالين** يستدعيان الوظيفة مرتين، أولاً عند تحميل الصفحة، ومرة أخرى عندما ينقر المستخدم على زر

يستخدم ال **(مثال)** الأول دالة عادية، ويستخدم ال **(مثال)** الثاني دالة مختصرة

توضح النتيجة أن ال **مثال** الأول يُرجع كائنين مختلفين (النافذة والزر)، بينما يُرجع **المثال** الثاني كائن النافذة مرتين، لأن كائن النافذة هو "مالك" الدالة

(**مثال** ✓ ↓↓)

يمثل الكائن الذي يستدعي الوظيفة **this** مع وظيفة عادية

```
// Regular Function:
Easy-to = function() {
  document.getElementById("Habib").innerHTML += this;
}

// The window object calls the function:
window.addEventListener("load", Easy-to);

// A button object calls the function:
document.getElementById("HABib_btn").addEventListener("click", Easy-to);
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

(**مثال** ✓ ↓↓)

تمثل منشأ الوظيفة **this** مع دالة مختصرة

```
// Arrow Function:
Easy-to = () => {
  document.getElementById("Habib").innerHTML += this;
}

// The window object calls the function:
window.addEventListener("load", Easy-to);
```

// A button object calls the function:

```
document.getElementById("HAbib_btn").addEventListener("click", Easy-to);
```

كتاب الكافي في جافاسكريبت الجزء الثاني ابو حبيب الحسيني

تذكر هذه الاختلافات عند العمل مع الوظائف. في بعض الأحيان يكون سلوك الوظائف العادية هو ما تريده، وإذا لم يكن الأمر كذلك، فاستخدم وظائف الأمثلة.

بناء الكلاسات

. لإنشاء كلاس **class** استخدم الكلمة المحجوزة

constructor(): أضف دائمًا وظيفة باسم

بناء الجملة

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

(مثال ↓↓)

```
class $Habib{  
  constructor(Name, year) {  
    this.Name = Name;  
    this.year = year;  
  }  
}
```

"\$Habib" يقوم ال (مثال ↓↓) أعلاه بإنشاء كلاس تسمى

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

. كلاس جافاسكريبت ليست كائنًا

. إنه قالب لكائنات جافاسكريبت

كيف استخدام الكلاس المنشأ

عندما يكون لديك كلاس ، يمكنك استخدامه لإنشاء كائنات

(مثال ✓)

```
let my$Habib1 = new $Habib("Habib", 2014);  
let my$Habib2 = new $Habib("Omar ibn Al-khattab", 2019);
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

• يستخدم ال (✓ مثال) أعلاه كلاس لإنشاء كائنين قيمة

• يتم استدعاء أسلوب المنشئ تلقائيًا عند إنشاء كائن جديد

وظيفة البناء للكلاس أو الكلاس

وظيفة البناء هي وظيفة خاصة

- "يجب أن يكون له الاسم الدقيق" المنشئ
- يتم تنفيذه تلقائيًا عند إنشاء كائن جديد
- يتم استخدامه لتهيئة خصائص الكائن

• إذا لم تحدد وظيفة منشئة، فستضيف جافاسكريبت وظيفة منشئة فارغة

طرق الكلاس

• يتم إنشاء دوال الكلاس أو الكلاس بنفس بناء الجملة مثل دوال الكائن

• لإنشاء كلاس **class** استخدم الكلمة المحجوزة

• وظيفة **constructor()** قم دائمًا بإضافة

• ثم أضف أي عدد من الطرق

بناء الجملة

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

```
method_1() { ... }
method_2() { ... }
method_3() { ... }
}
```

: أنشئ وظيفة كلاس باسم "حبيب"، والتي تُرجع الاسم والسنة

(مثال ↓↓↓)

```
class $Habib{
  constructor(Name, year) {
    this.Name = Name;
    this.year = year;
  }
  age() {
    let date = new Date();
    return date.getFullYear() - this.year;
  }
}
```

```
let my$Habib= new $Habib("Habib", 2014);
document.getElementById("Habib").innerHTML =
"My $Habibis " + my$Habib.age() + " years olabibDate.";
```

كتاب الكافي في جافاسكريبت الجزء الثاني أبو حبيب الحسيني

:يمكنك إرسال البرمجات إلى دوال الفصل او الكلاس

(مثال ↓↓↓)

```
class $Habib{
  constructor(Name, year) {
    this.Name = Name;
    this.year = year;
  }
  age(x) {
    return x - this.year;
  }
}
```

```
let date = new Date();
let year = date.getFullYear();
```



```
let my$Habib= new $Habib("Habib", 2014);
document.getElementById("Habib").innerHTML=
"My $Habibis " + my$Habib.age(year) + " years olabibDate.";
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

وحدات جافاسكربت

الوحدات

تسمح لك وحدات جافاسكربت بتقسيم ال اكواد إلى ملفات منفصلة . وهذا يجعل من السهل الحفاظ على قاعدة ال اكواد **import** و **export** تعتمد وحدات جافاسكربت على العبارات

كيف تصدير الجبرانات

يمكنك تصدير دالة أو متغير من أي ملف .ونملأه بالأشياء التي نريد تصديرها **hAbiB.js** دعونا ننشئ ملفاً باسم .هناك نوعان من عمليات التصدير: المحددة والافتراضية

تسمية التصدير

يمكنك إنشاء عمليات تصدير محددة بطريقتين. في الخط بشكل فردي، أو كلها مرة واحدة في الأسفل

في الخط بشكل فردي:

hAbiB.js

```
export const Name = "hAbiB";  
export const age = 40;
```

الكلمة واحدة في الأسفل:

hAbiB.js

```
const Name = "hAbiB";  
const age = 40;  
  
export {Name, age};
```

التصدير الافتراضي

ونستخدمه لتوضيح التصدير الافتراضي، **habib.js** دعونا ننشئ ملفاً آخر، اسمه **habib.js**. يمكنك الحصول على تصدير افتراضي واحد فقط في الملف.

(مثال ↓↓↓)

habib.js

```
const message = () => {  
  const Name = "hAbiB";  
  const age = 40;  
  return Name + ' is ' + age + 'years olabibDate!';  
};  
  
export default message;
```

كيف الاستيراد

يمكنك استيراد الوحدات الخاصة إلى ملف بطريقتين، بناءً على ما إذا كانت تسمى التصديرات أو الاصلرات الافتراضية. يتم إنشاء التصديرات المحددة باستخدام الأقواس المتعرجة. التصديرات الافتراضية ليست كذلك.

كيف الاستيراد من التصديرات المنشأة

استيراد الاجراءات المحددة من الملف hAbiB.js:

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

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

الاستيراد من التصديرات الافتراضية

استيراد تصدير افتراضي من الملف habib.js:

```
import message from "./habib.js";
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

ملحوظة

HTTP(s) تعمل الوحدات فقط مع بروتوكول

جافاسكربت جيسون

ما هو جيسون؟

.

- JSON هو تنسيق قاعدة بيانات خفيف الوزن
- JSON * مستقلة عن اللغة
- JSON سهل الفهم

جيسون (مثال)

كائن الموظفين: مصفوفة من 3 سجلات موظفين (كائنات) JSON تحدد صيغة

جيسون (↓↓↓ مثال ✓)

```
{
  "employees": [
    {"firstName": "Habib", "lastName": "Al Husini 🇸🇩🇸🇩🇸🇩"},
    {"firstName": "Mohssen", "lastName": "Hassan"},
    {"firstName": "Habib", "lastName": "Fareed"}
  ]
}
```

لكائنات JSON يتم تقييم تنسيق

من الناحية النحوية مع ال اكواد لإنشاء كائنات جافاسكربت JSON يتطابق تنسيق

بسهولة إلى كائنات جافاسكربت أصلية JSON وبسبب هذا التشابه، يمكن لبرنامج جافاسكربت تحويل كلمات

قواعد بناء جملة JSON

- يتم فصل بين القيمة والمفتاح بفواصل
- الأقواس المتعرجة تجزء مجموعات القيم والمفاتيح
- الأقواس المربعة تحمل المصفوفات

اسم وقيمة - JSON كلمات

كأزواج اسم/قيمة، تمامًا مثل خصائص كائن جافاسكربت JSON تتم كتابة كلمات

يتكون زوج الاسم/القيمة من اسم الحقل (بين علامتي اقتباس مزدوجتين)، متبوعًا بنقطتين، متبوعة بقيمة

```
"firstName":"Habib"
```

JSON كائنات

داخل الأقواس المتعرجة JSON تتم كتابة كائنات

تمامًا كما هو الحال في جافاسكربت، يمكن أن تحتوي الكائنات على أزواج متعددة من الأسماء/القيم

```
{"firstName":"Habib", "lastName":"Al Husini 😊😊😊"}
```

مصفوفات جيسون

بين قوسين مربعين JSON تتم كتابة مصفوفات

تمامًا كما هو الحال في جافاسكربت، يمكن أن تحتوي المصفوفة على كائنات

```
"employees":[
```

```
  {"firstName":"Habib", "lastName":"Al Husini 😊😊😊"},
```

```
  {"firstName":"Mohssen", "lastName":"Hassan"},
```

```
  {"firstName":"Habib", "lastName":"Fareed"}
```

```
]
```

في ال (✓ مثال ↓↓) أعلاه، الكائن "المستخدمين" عبارة عن مصفوفة. أنه يحتوي على ثلاثة كائنات

كل كائن هو سجل لعنصر (بالاسم الأول واسم العائلة)

تحويل JSON

هو قراءة الكلمات من خادم الويب وعرض الكلمات في صفحة الويب JSON الاستخدام الشائع لـ للتبسيط، يمكن إثبات ذلك باستخدام نص كمدخل

JSON: أولاً، أنشئ نص جافاسكربت تحتوي على صيغة

```
let abib = '{ "employees" : [ +  
{ "firstName": "Habib" , "lastName": "Al Husini 🤪🤪🤪" }, +  
{ "firstName": "Mohssen" , "lastName": "Hassan" }, +  
{ "firstName": "Habib" , "lastName": "Fareed" } ] }';
```

لتحويل النص إلى كائن جافاسكربت JSON.parse() بعد ذلك، استخدم وظيفة جافاسكربت المضمنة

```
const obj = JSON.parse(abib);
```

وأخيراً، استخدم كائن جافاسكربت الجديد في صفحتك

(مثال ↓↓)

```
<p id="Habib"></p>  
<script>  
document.getElementById("Habib").innerHTML =  
obj.employees[1].firstName + " " + obj.employees[1].lastName;  
</script>
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

تصحيح أخطاء جافاسكربت

يمكن أن تحدث أخطاء في كل مرة تكتب فيها بعض اكواد الكمبيوتر الجديدة

console.log() وظيفة

لعرض قيم جافاسكربت في نافذة مصحح `console.log()` إذا كان المتصفح يدعم تصحيح الأخطاء، فيمكنك استخدامه للأخطاء:

(مثال ↓↓↓)

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

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

<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>

</body>
</html>
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

تحديد نقاط التوقف

في نافذة مصحح الأخطاء، يمكنك تعيين نقاط التوقف في كود جافاسكربت عند كل نقطة توقف، سيتوقف تنفيذ جافاسكربت، ويتيح لك فحص قيم جافاسكربت بعد فحص القيم، يمكنك استئناف تنفيذ الكواد (عادةً باستخدام زر التشغيل).

الكلمة المحجوزة للمصحح

المحجوزة تنفيذ الكود ، وتستدعي وظيفة تصحيح الأخطاء (إذا كانت متوفرة) **debugger** توقف الكلمة هذا له نفس وظيفة تحديد نقطة توقف في مصحح الأخطاء. في حالة عدم توفر أي تصحيح، لن يكون لعبارة مصحح الأخطاء أي تأثير. عند تشغيل مصحح الأخطاء، سيتوقف تنفيذ هذا الرمز قبل تنفيذ السطر الثالث.

(مثال ↓↓↓)

```
let x = 15 * 5;  
debugger;  
document.getElementById("Habib").innerHTML = x;
```

كتاب الكافي في جافاسكربت الجزء الثاني ابو حبيب الحسيني

هل كنت تعلم؟

تصحيح الأخطاء هو عملية اختبار الأخطاء (الأخطاء) في برامج الكمبيوتر وإيجادها وتقليلها. كان أول خطأ حاسوبي معروف هو خلل حقيقي (حشرة) عالق في الأجهزة الإلكترونية.

ملحوظة

ضع دائماً مسافات حول عوامل التشغيل (= + * /) وبعد الفواصل

أمثلة:

```
let x = y + z;  
const Husinie_Array = ["Hamza", "Habib", "Mohamed"];
```

المسافة البادئة للكود

: استخدم دائمًا مسافتين لوضع مسافة بادئة لكنتل ال اكواد

المهام:

```
functionhapip(bib) {  
  return (5 / 9) * (bib - 32);  
}
```

للمسافة البادئة. يقوم المحررون المختلفون بتفسير علامات التبويب بشكل (tableators) لا تستخدم علامات التبويب مختلف.

قواعد

القواعد العامة للعبارات البسيطة:

- قم دائمًا بإنهاء العبارة البسيطة بفاصلة منقوطة.

أمثلة:

```
const HaBiB= ["Hamza", "Habib", "Mohamed"];
```

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

القواعد العامة للكلمات المعقدة (المركبة):

- ضع قوس الفتح في نهاية السطر الأول.

- استخدم مسافة واحدة قبل قوس الفتح.
- ضع قوس الإغلاق على سطر جديد، بدون مسافات بادئة.
-

المهام:

```
function hapip(bib) {  
  return (5 / 9) * (bib - 32);  
}
```

طرق تهيئة المتغيرات والبيانات

```
// Declare and initiate at the beginning  
let firstName = "";  
let lastName = "";  
let hAbiB3 = 0;  
let discount = 0;  
let fullhAbiB3 = 0,  
const Husinie_Array = [];  
const myObject = {};
```

توفر تهيئة المتغيرات فكرة عن الاستخدام المقصود (ونوع الكلمات المقصودة).

الطرق البديلة

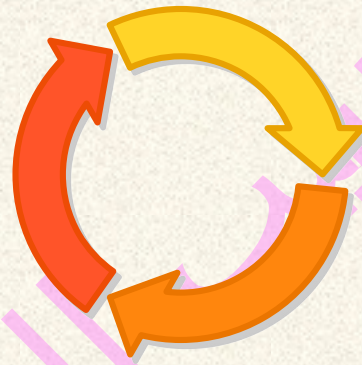
- `new String()` استخدم "" بدلا من
- `new Number()` استخدم 0 بدلا من
- `new Boolean()` بدلا من `false` استخدم
- `new Object()` استخدم {} بدلا من
- `new Array()` استخدم [] بدلا من
- `new RegExp()` استخدم /()/ بدلا من
- `new Function()` بدلا من `function () {}` استخدم

(↓↓↓ مثال ✓)

```
let x1 = ""; // new primitive string
let x2 = 0; // new primitive number
let x3 = false; // new primitive boolean
const x4 = {}; // new object
const x5 = []; // new array object
const x6 = /()/; // new regexp object
const x7 = function(){}; // new function object
```

كتاب الكافي في جافاسكربت الجزء الثاني أبو حبيب الحسيني

أبو حبيب الحسيني



التكملة في الجزء الثالث

بإذن الله تعالى