NodeJs

بالعربى

ابو حبيب الحسينى

يا امة الاسـلام فـــــــجرك نورا

والروض فى ساحات مجدك اظـهرا

سحب المعالى فى سمائك امطرت

غيثا واجرت فى رحابك انـــــــهرا

لبسة بها الاشـــــــجار ثوبا مورقا

وغدت بها الصـــــحراء روضا اخضرا

سنحيـــــا فى ظل الـــعقيدة امـــتنا

مرفوعـة الاعلام محكـــــــمة العرى

فيا امة الاســـــــلام فجـــرك نورا

الفهرس

مقدمة قصيرة

بسم الله الرحمن الرحيم

النود جى اس بختصار هو اطار عمل مجانى مفتوح المصدر تمت كتابة بلغة السي بلاص يعمل على جميع المنصات والانظمة المختلفة مثل ويندوز ولينكس وماك وغيرها متعدد الاستخدامات وهذا الاطار بمثابة استقلاليه لجافاسربت ومن مميزات نود جى اس تصنيفة بانة اسرع ريكويست فى العالم يصل الى 600 ريكويست فى الثانية الواحدة وهذا اسرع من بى اتش بى عشر مرات وايضا صنف مستودع حزم ان بى ام الذى يخدم نود جى اس انه اكبر مستودع حزمة برجية فى العالم وهذا يرجع الى شهرة جافا سكربت

وعدد مستخدميها الذى لا حصر له حتى على موقع الجيتهاب جافا سكربت هى الاولى عالميا بعد بايثون فقد تخطة مشاريع جافاسكربت على موقع الجيتهاب 2 مليون لتصبح جافاسكربت فى المركز الاول عالميا من حيث عدد المشاريع ثم تاتى بى اتش بى فى المركز الثانى بمجموعة مشاريع مليون وكسور فاذا دخلت على موقع الجيتهاب وكتب فى خانة البحث جافاسكربت فستجد الرقم الرهيب لعدد المستودعات الذى تحطى 2 مليون وهو ثانى اكبر عدد موجود حتى الان بعد بايثون و فى اذدياد مستمر ومن مميزات النود جى اس ايضا انك تستطيع استخدام جافاسكربت خارج المتصفح مثل اى لغة برمجة عادية

تنشا تطبيقات لانظمة التشغيل المختلفة سواء موبايل او كمبيوتر وايضا تستطيع دمج تقنيات الويب العادية من اتش تى ام ال وسي اس اس والمكتبات المعروفة مثل الجيكويرى وغيرها فى مشاريعك على انظمة التشغيل المختلفة وايضا من المميزات الرهيب لهذا الاطار ان كود جافا سكربت ثابت على كل انظمة التشغيل التى تقوم بنشاء تطبيقات لها على خلاف اللغات الاخرى التى تعتمد لكل نظام تشغيل مكتبات خاصة به التى يجب ان تستدعيها لتعمل عليها مما يجعل الامر معقد للغاية ناهيك على ان نود جى اس فى الاساس هو بيئة خادم مفتوحة المصدر. يسمح لك بتشغيل جافاسكربت على الخادم لتعمل مثل بى اتش بى تمام

مثال التكوين الاساسى

هذا مثال لتكوين السيرفر الاساسى الذى ستعمل علية نلاحظ ان الكود قصير جدا وهذا ايضا من مميزات هذا اطار انه يختصر الاكواد

مثال

```
var http = require('http');

http.createServer(function (req, res) {
```

```
  res.writeHead(200, {'Content-
Type': 'text/plain'});
  res.end('Abo Habib Al_Hosiny !);
}).listen(8080);
```

أمثلة على التشغيل في واجهة سطر الأوامر

في هذا الكتاب ، ستكون هناك بعض الأمثلة التي سيتم شرحها بشكل أفضل من خلال عرض النتيجة في واجهة سطر الأوامر.

عندما يحدث ذلك، ستعرض أداة "اظهار نود جي اس " النتيجة في شاشة سوداء على اليمين:

مثال

```
console.log('Abo Habib Al_Hosiny !');
console.log('The result is displayed in the
Command Line Interface');
```

الإستخدامات المميزة كخادم ويب

يستخدم برمجة غير متزامنة! و

يمكن أن تكون المهمة الشائعة لخادم الويب هي فتح ملف على الخادم وإعادة المحتوى إلى العميل. وهذا يتم فى سرعة كبيرة جدا مقارنتا بالخوادم الاخرى

مع طلب الملف ASP أو PHP إليك كيفية تعامل:

1. يرسل المهمة إلى نظام ملفات الكمبيوتر.

2. ينتظر بينما يفتح نظام الملفات ويقرأ الملف.
3. إرجاع المحتوى إلى العميل.
4. على استعداد للتعامل مع الطلب التالي.

إليك كيفية تعامل نود جى اس مع طلب الملف:

1. يرسل المهمة إلى نظام ملفات الكمبيوتر.
2. على استعداد للتعامل مع الطلب التالي.
3. عندما يقوم نظام الملفات بفتح الملف وقراءته، يقوم الخادم بإرجاع المحتوى إلى العميل.
4. دمج اكواد الباكاند مع الفرونت اند وهذا ما لم تستطع فعلة فى الخوادم الاخرى

يزيل الانتظار، ويستمر ببساطة في الطلب التالي.

يقوم بتشغيل برمجة أحادية الترابط وغير محظورة وغير متزامنة، وهي فعالة وسريعة جدًا في استخدام الذاكرة.

- يمكن لـ نود جى اس إنشاء محتوى صفحة ديناميكي
- يمكن لـ نود جى اس إنشاء الملفات وفتحها وقراءتها وكتابتها وحذفها وإغلاقها على الخادم
- يمكن لـ نود جى اس جمع بيانات النموذج

• يمكن لـ نود جى اس إضافة أو حذف أو تعديل البيانات في قاعدة البيانات

• تحتوي ملفات على المهام التي سيتم تنفيذها في أحداث معينة
• الحدث النموذجي هو شخص يحاول الوصول إلى منفذ على الخادم
• يجب أن يتم تشغيل ملفات على الخادم قبل أن يكون لها أي تأثير
• ملفات لها الامتداد **".js"**

تحميل

يحتوي الموقع الرسمي على تعليمات تثبيت نود جى اس
: https://nodejs.org

ابدء الان

بمجرد قيامك بتنزيل وتثبيته على جهاز الكمبيوتر، فلنحاول عرض "Abo Habib Al_Hosiny !" في متصفح الويب.

أنشئ ملف نود جى اس باسم "file_Abo_Habib.js".
وأضف الكود التالي:

file_Abo_Habib.js

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-
Type': 'text/html'});
  res.end('Abo Habib Al_Hosiny !');
}).listen(8080);
```

احفظ الملف على جهاز الكمبيوتر:

C:\Users\ Your Name \file_Abo_Habib.js

يطلب الكود من الكمبيوتر أن يكتب "Abo Habib Al_Hosiny !!" إذا حاول أي شخص (مثل متصفح الويب) الوصول إلى جهاز الكمبيوتر على المنفذ 8080.

في الوقت الحالي، ليس عليك فهم الكود. سيتم شرحه لاحقا.

واجهة سطر الأوامر

يجب أن يتم تشغيل ملفات نود جى اس في برنامج "Command Line Interface" بجهاز الكمبيوتر.

تعتمد كيفية فتح واجهة سطر الأوامر على جهاز الكمبيوتر على نظام التشغيل. بالنسبة لمستخدمي Windows، اضغط على زر البداية في حقل "cmd" وابحث عن "موجه الأوامر"، أو ببساطة اكتب البحث.

انتقل إلى المجلد الذي يحتوي على الملف "file_Abo_Habib.js"، يجب أن تبدو نافذة واجهة سطر الأوامر كما يلي:

```
C:\Users\Your Name>_
```

قم بتشغيل ملف

يجب أن يتم تشغيل الملف الذي قمت بإنشائه للتو بواسطة نود جى اس قبل اتخاذ أي إجراء.

ابدأ واجهة سطر الأوامر، ثم اكتب node file_Abo_Habib.js
ثم اضغط على زر الإدخال:

بدء "file_Abo_Habib.js":

```
C:\Users\Your Name>node file_Abo_Habib.js
```

الآن، جهاز الكمبيوتر يعمل كخادم!

إذا حاول أي شخص الوصول إلى جهاز الكمبيوتر عبر المنفذ 8080، فسوف يحصل على رسالة "Abo Habib Al_Hosiny !!" رسالة في المقابل!

قم بتشغيل متصفح الإنترنت، واكتب العنوان:

http://localhost:8080

ما هي الوحدة النمطية ؟

ضع في اعتبارك أن الوحدات النمطية هي نفس مكتبات جافاسكربت.

مجموعة من الوظائف التي تريد تضمينها في التطبيق.

وحدات مدمجة

يحتوي نود جى اس على مجموعة من الوحدات المضمنة التي يمكنك استخدامها دون أي تثبيت إضافي.

لدينا للحصول على قائمة كاملة بالوحدات.

تضمين الوحدات

الدالة التي تحمل اسم require() لتضمين وحدة، استخدم الوحدة:

```
var http = require('http');
```

وأصبح قادرًا HTTP. الآن أصبح لتطبيقك حق الوصول إلى وحدة على إنشاء خادم:

```
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-
Type': 'text/html'});
  res.end('Abo Habib Al_Hosiny !');
}).listen(8080);
```

إنشاء الوحدات

يمكنك إنشاء الوحدات النمطية وإدراجها بسهولة في تطبيقاتك.

يقوم المثال التالي بإنشاء وحدة تقوم بإرجاع كائن التاريخ والوقت:

مثال

قم بإنشاء وحدة تقوم بإرجاع التاريخ والوقت الحاليين:

```
exports.Hosini_DateTime = function () {
  return Date();
};
```

الكلمة الأساسية لتوفير الخصائص والأساليب **exports** استخدم خارج ملف الوحدة النمطية.

احفظ الكود أعلاه في ملف يسمى
"Module_Abo_Habib.js"

قم بتضمين الوحدة

يمكنك الآن تضمين الوحدة واستخدامها في أي من ملفات نود جى اس .

مثال

في ملف نود جى "Module_Abo_Habib" استخدم الوحدة اس :

```
var http = require('http');
var dt = require('./Module_Abo_Habib');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-
Type': 'text/html'});
  res.write("The date and time are currently:
" + dt.Hosini_DateTime());
  res.end();
}).listen(8080);
```


لاحظ أننا نستخدم ./لتحديد موقع الوحدة، وهذا يعني أن الوحدة موجودة في نفس المجلد مثل ملف نود جى اس .

احفظ الكود أعلاه في ملف يسمى "Habib_module.js"، وابدأ الملف:

بدء Habib_module.js:

```
C:\Users\Your Name>node Habib_module.js
```

إذا قمت بإتباع نفس الخطوات على جهاز الكمبيوتر، فسوف ترى نفس النتيجة كما في المثال:

http://localhost:8080

وحدة HTTP

وحدة HTTP

والتي HTTP، يحتوي نود جى اس على وحدة مدمجة تسمى تسمح لـ نود جى اس بنقل البيانات عبر بروتوكول نقل النص التشعبي (HTTP).

الطريقة require() استخدم HTTP، لتضمين وحدة:

```
var http = require('http');
```

نود جى اس كخادم ويب

يستمع إلى منافذ HTTP إنشاء خادم HTTP يمكن لوحدة الخادم ويعطي استجابة للعميل.

الطريقة لإنشاء خادم HTTP: استخدم createServer()

مثال

```
var http = require('http');

//create a server object:
```

```
http.createServer(function (req, res) {
  res.write('Abo Habib Al_Hosiny !); //write a
response to the client
  res.end(); //end the response
}).listen(8080); //the server object listens on
port 8080
```


http.createServer() سيتم تنفيذ الوظيفة التي تم تمريرها إلى الطريقة عندما يحاول شخص ما الوصول إلى الكمبيوتر على المنفذ 8080.

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_http.js"، ثم ابدأ الملف:

بدء المثال

```
C:\Users\Your Name>node
Abo_Habib_http.js
```

إذا قمت بإتباع نفس الخطوات على جهاز الكمبيوتر، فسوف ترى نفس النتيجة كما في المثال:

http://localhost:8080

أضف اتش تى ام ال

إذا كان من المفترض أن يتم عرض الاستجابة من خادم HTTP بنوع HTTP فيجب عليك تضمين رأس، HTML بتنسيق المحتوى الصحيح:

مثال

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200,

{'ContentType': 'text/html'});
  res.write('Abo Habib Al_Hosiny !);
  res.end();
}).listen(8080);
```

هي رمز الحالة، 200 res.writeHead() الوسيطة الأولى للطريقة يعني أن كل شيء على ما يرام، والوسيطة الثانية هي كائن يحتوي على رؤوس الاستجابة.

قراءة سلسلة الاستعلام

http.createServer() تحتوي الدالة التي تم تمريرها كائن) وسيطة تمثل الطلب من العميل، ككائن req إلى http.IncomingMessage).

والتي تحتوي "url" يحتوي هذا الكائن على خاصية تسمى الذي يأتي بعد اسم المجال url على جزء عنوان:

Abo_Habib_http_url.js

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200,
{'ContentType': 'text/html'});
  res.write(req.url);
```

```
  res.end();
}).listen(8080);
```

احفظ الكود أعلاه في ملف يسمى
وابدأ الملف "Abo_Habib_http_url.js"

بدء المثال

```
C:\Users\Your Name>node
Abo_Habib_http_url.js
```

إذا اتبعت نفس الخطوات على جهاز الكمبيوتر، فيجب أن ترى
نتيجتين مختلفتين عند فتح هذين العنوانين:

تقسيم سلسلة الاستعلام

توجد وحدات مدمجة لتقسيم سلسلة الاستعلام بسهولة إلى أجزاء قابلة للقراءة، مثل وحدة URL.

مثال

قم بتقسيم سلسلة الاستعلام إلى أجزاء قابلة للقراءة:

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-
Type': 'text/html'});
  var q = url.parse(req.url, true).query;
  var txt = q.year + " " + q.month;
  res.end(txt);
}).listen(8080);
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_querystring.js" وابدأ الملف:

بدء Abo_Habib_querystring.js:

```
C:\Users\Your Name>node
Abo_Habib_querystring.js
```

العنوان:

http://localhost:8080/?year=2017&month=July

سوف تنتج هذه النتيجة:

```
2017 July
```


وحدة نظام الملفات

التحكم فى الملفات

تسمح لك وحدة نظام الملفات نود جى اس بالعمل مع نظام الملفات على جهاز الكمبيوتر.

الطريقة require() لتضمين وحدة نظام الملفات، استخدم:

```
var fs = require('fs');
```

الاستخدام الشائع لوحدة نظام الملفات:

- قراءة الملفات
- إنشاء ملفات
- تحديث الملفات
- حذف الملفات
- إعادة تسمية الملفات
- وغير الكثير والكثير من اجرائات الملفات

قراءة الملفات

لقراءة الملفات الموجودة fs.readFile() يتم استخدام الطريقة على جهاز الكمبيوتر.

التالي (الموجود في نفس المجلد HTML افترض أن لدينا ملف
مثل نود جى اس):

Abo_Habib_File.html

```
<html>
<body>
<h1>Hosini_ Header</h1>
<p>Hosini_ paragraph.</p>
</body>
</html>
```

أنشئ ملف نود جى اس الذي يقرأ ملف HTML. ويعيد
المحتوى:

مثال

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {

fs.readFile('Abo_Habib_File.html', function(e
```

```
rr, data) {
    res.writeHead(200, {'Content-
Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_readfile.js"، وابدأ الملف

بدء Abo_Habib_readfile.js:

```
C:\Users\Your Name>node
Abo_Habib_readfile.js
```

إذا قمت بإتباع نفس الخطوات على جهاز الكمبيوتر، فسوف ترى نفس النتيجة كما في المثال:

http://localhost:8080

إنشاء ملفات

تحتوي وحدة نظام الملفات على طرق لإنشاء ملفات جديدة:

- fs.appendFile()
- fs.open()
- fs.writeFile()

تقوم الطريقة fs.appendFile() بإلحاق محتوى محدد بملف. إذا كان الملف غير موجود، سيتم إنشاء الملف:

مثال

قم بإنشاء ملف جديد باستخدام طريقة appendFile():

```
var fs = require('fs');

fs.appendFile('Abo_Habib_File1.txt', 'Hello
content!', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

علامة" كوسيطة ثانية، إذا كانت "fs.open()" تأخذ الطريقة لـ "الكتابة"، فسيتم فتح الملف المحدد للكتابة. إذا "w" العلامة كان الملف غير موجود، يتم إنشاء ملف فارغ:

مثال

قم بإنشاء ملف جديد فارغ باستخدام طريقة open() :

```
var fs = require('fs');

fs.open('Abo_Habib_File2.txt', 'w', function (
err, file) {
  if (err) throw err;
  console.log('Saved!');
});
```

الملف والمحتوى المحددين إذا fs.writeFile() تستبدل الطريقة كان موجودًا. في حالة عدم وجود الملف، سيتم إنشاء ملف جديد يحتوي على المحتوى المحدد:

مثال

قم بإنشاء ملف جديد باستخدام طريقة writeFile():

```
var fs = require('fs');

fs.writeFile('Abo_Habib_File3.txt', 'Hello
content!', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

تحديث الملفات

تحتوي وحدة نظام الملفات على طرق لتحديث الملفات:

- fs.appendFile()
- fs.writeFile()

fs.appendFile() تُلحق الطريقة المحتوى المحدد في نهاية الملف المحدد:

مثال

إلحاق "هذا هو النص." إلى نهاية الملف "Abo_Habib_File1.txt":

```
var fs = require('fs');
```

```
fs.appendFile('Abo_Habib_File1.txt', ' This is
Hosini_ text.', function (err) {
  if (err) throw err;
  console.log('Updated!');
});
```

تستبدل الطريقة fs.writeFile() الملف والمحتوى المحددين:

مثال

استبدال محتوى الملف "Abo_Habib_File3.txt":

```
var fs = require('fs');

fs.writeFile('Abo_Habib_File3.txt', 'This is
Hosini_ text', function (err) {
  if (err) throw err;
  console.log('Replaced!');
});
```

حذف الملفات

لحذف ملف باستخدام وحدة نظام الملفات، استخدم الطريقة fs.unlink().

الطريقة fs.unlink() تحذف الملف المحدد:

مثال

حذف "Abo_Habib_File2.txt":

```
var fs = require('fs');

fs.unlink('Abo_Habib_File2.txt', function (err
) {
  if (err) throw err;
  console.log('File deleted!');
});
```

إعادة تسمية الملفات

لإعادة تسمية ملف باستخدام وحدة نظام الملفات، استخدم fs.rename() الطريقة.

تقوم الطريقة fs.rename() بإعادة تسمية الملف المحدد:

مثال

إعادة تسمية "Abo_Habib_File1.txt" إلى "Al_Hosiny_File.txt":

```
var fs = require('fs');

fs.rename('Abo_Habib_File1.txt', 'Al_Hosiny_
File.txt', function (err) {
  if (err) throw err;
```

```
  console.log('File Renamed!');
});
```

وحدة URL

تقوم وحدة **URL** بتقسيم عنوان الويب إلى أجزاء قابلة للقراءة.

الطريقة **require()** استخدم **URL**، لتضمين وحدة:

```
var url = require('url');
```

الطريقة، وستعيد **url.parse()** قم بتحليل عنوان باستخدام كائن **URL** مع كل جزء من العنوان كخصائص:

مثال

تقسيم عنوان الويب إلى أجزاء قابلة للقراءة:

```
var url = require('url');
var adr
= 'http://localhost:8080/Al_Hosiny.htm?
year=2017&month=february';
var q = url.parse(adr, true);

console.log(q.host); //returns
'localhost:8080'
console.log(q.pathname); //returns
'/Al_Hosiny.htm'
console.log(q.search); //returns '?
year=2017&month=february'

var qdata = q.query; //returns an object:
{ year: 2017, month: 'february' }
console.log(qdata.month); //returns
'february'
```

خادم الملفات

الآن نحن نعرف كيفية تحليل سلسلة الاستعلام، وفي الفصل السابق تعلمنا كيفية جعل نود جى اس يعمل كخادم ملفات. دعونا نجمع بين الاثنين ونخدم الملف الذي طلبه العميل.

واحفظهما في نفس المجلد مثل html قم بإنشاء ملفين بتنسيق ملفات نود جى اس .

Summer.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Summer</h1>
<p>I love the Al Hosini!</p>
```

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

Winter.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Winter</h1>
<p>I love the Al Hosini !</p>
</body>
</html>
```

قم بإنشاء ملف نود جى اس الذي يفتح الملف المطلوب ويعيد المحتوى إلى العميل. إذا حدث أي خطأ، قم بإلقاء خطأ 404:

Abo_Habib_fileserver.js:

```
var http = require('http');
var url = require('url');
var fs = require('fs');

http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename = "." + q.pathname;
  fs.readFile(filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-
Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-
Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

تذكر أن تبدأ الملف:

بدء Abo_Habib_fileserver.js:

```
C:\Users\Your Name>node
Abo_Habib_fileserver.js
```

إذا اتبعت نفس الخطوات على جهاز الكمبيوتر، فيجب أن ترى نتيجتين مختلفتين عند فتح هذين العنوانين:

NPM

ما هو NPM؟

هو مدير حزم لحزم نود جى اس ، أو الوحدات النمطية إذا **NPM** أردت.

آلاف الحزم المجانية للتنزيل **www.npmjs.com** يستضيف والاستخدام

على جهاز الكمبيوتر عند تثبيت نود **NPM** يتم تثبيت برنامج جى اس

NPM جاهز بالفعل للتشغيل على جهاز الكمبيوتر!

ما هي الحزمة؟

تحتوي الحزمة الموجودة في نود جى اس على كافة الملفات التي تحتاجها للوحدة النمطية.

يمكنك تضمينها في JavaScript الوحدات هي مكتبات مشروعك.

قم بتنزيل الحزمة

تنزيل الحزمة سهل للغاية.

افتح واجهة سطر الأوامر واطلب من NPM تنزيل الحزمة التي تريدها.

أريد تنزيل حزمة تسمى "الأحرف الكبيرة":

```
C:\Users\Your Name>npm install upper-case
```

لقد قمت الآن بتنزيل وتثبيت الحزمة الأولى !

حيث ، "Hosini_modules" بإنشاء مجلد باسم NPM يقوم
سيتم وضع الحزمة. سيتم وضع جميع الحزم التي تقوم بتثبيتها
في المستقبل في هذا المجلد.

يحتوي مشروعي الآن على بنية مجلد مثل هذا:

C:\Users\Hosini_ Name\Hosini_modules\upper-case

باستخدام الحزمة

بمجرد تثبيت الحزمة، فهي جاهزة للاستخدام.

قم بتضمين الحزمة "ذات الأحرف الكبيرة" بنفس الطريقة التي
تقوم بها بتضمين أي وحدة أخرى:

```
var uc = require('upper-case');
```

قم بإنشاء ملف نود جى اس الذي سيقوم بتحويل الإخراج "Abo
Habib Al_Hosiny !!" إلى أحرف كبيرة:

مثال

```
var http = require('http');
var uc = require('upper-case');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-
Type': 'text/html'});
  res.write(uc.upperCase("Abo Habib
Al_Hosiny !!"));
  res.end();
}).listen(8080);
```


احفظ الكود أعلاه في ملف يسمى "Abo_Habib_uppercase.js"، وابدأ الملف:

بدء المثال للأحرف الكبيرة:

```
C:\Users\Your Name>node
Abo_Habib_uppercase.js
```

إذا قمت بإتباع نفس الخطوات على جهاز الكمبيوتر، فسوف ترى نفس النتيجة كما في المثال:

http://localhost:8080

الإحداث

نود جى اس مثالي للتطبيقات التي تعتمد على الأحداث.

ما هى الأحداث

كل إجراء على جهاز الكمبيوتر هو حدث. كما هو الحال عند إجراء اتصال أو فتح ملف.

يمكن للكائنات في نود جى اس إطلاق الأحداث، مثل إطلاق الأحداث عند فتح ملف وإغلاقه **readStream** كائن:

مثال

```
var fs = require('fs');
var rs =
fs.createReadStream('./Abo_Habib_File.txt')
;
rs.on('open', function () {
  console.log('The file is open');
});
```

وحدة الأحداث

يحتوي نود جى اس على وحدة مدمجة، تسمى الأحداث، حيث يمكنك إنشاء الأحداث وتشغيلها والاستماع إليها.

الطريقة **require()** لتضمين وحدة الأحداث المضمنة، استخدم بالإضافة إلى ذلك، جميع خصائص الحدث والأساليب هي مثيل لكائن **EventEmitter.** لتتمكن من الوصول إلى هذه الخصائص والأساليب، قم بإنشاء كائن **EventEmitter:**

```
var events = require('events');
var eventEmitter
= new events.EventEmitter();
```

كائن EventEmitter

يمكنك تعيين معالجات الأحداث للأحداث باستخدام كائن **EventEmitter.**

في المثال أدناه قمنا بإنشاء دالة سيتم تنفيذها عند إطلاق حدث

الطريقة **emit()** لإطلاق حدث ما، استخدم.

مثال

```
var events = require('events');
var eventEmitter
= new events.EventEmitter();

//Create an event handler:
var Hosini_EventHandler = function () {
  console.log('I hear a scream!');
```

```
}

//Assign the event handler to an event:
eventEmitter.on('scream',
Hosini_EventHandler);

//Fire the 'scream' event:
eventEmitter.emit('scream');
```

رفع او تحميل الملفات

الوحدة التحميل والرفع

هناك وحدة جيدة جدًا للعمل مع تحميل الملفات تسمى **"Formidable".**

يمكن تنزيل الوحدة **Formidable** وتثبيتها باستخدام **NPM:**

```
C:\Users\Your Name>npm install formidable
```

يمكنك تضمين الوحدة Formidable، بعد قيامك بتنزيل وحدة في أي تطبيق:

```
var formidable = require('formidable');
```

أنت الآن جاهز لإنشاء صفحة ويب في نود جى اس تتيح للمستخدم تحميل الملفات إلى جهاز الكمبيوتر:

الخطوة 1: إنشاء نموذج تحميل

قم بإنشاء ملف نود جى اس الذي سيرفع الملف، مع HTML حقل التحميل:

مثال

سينتج عن هذا الكود نموذج HTML:

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-
Type': 'text/html'});
  res.write('<form
action="Hosini_file_upload" method="post"
enctype="multipart/form-data">');
  res.write('<input type="file"
name="filetoupload"><br>');
  res.write('<input type="submit">');
  res.write('</form>');
  return res.end();
}).listen(8080);
```


الخطوة 2: تحليل الملف الذي تم تحميله

لتتمكن من تحليل الملف Formidable قم بتضمين الوحدة الذي تم تحميله بمجرد وصوله إلى الخادم.

عندما يتم تحميل الملف وتحليله، يتم وضعه في مجلد مؤقت على جهاز الكمبيوتر.

مثال

سيتم تحميل الملف ووضعه في مجلد مؤقت:

```
var http = require('http');
var formidable = require('formidable');

http.createServer(function (req, res) {
  if (req.url == '/Hosini_file_upload') {
    var form
= new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      res.write('File uploaded');
      res.end();
    });
  } else {
    res.writeHead(200, {'Content-
Type': 'text/html'});
    res.write('<form
```

```
action="Hosini_file_upload" method="post"
enctype="multipart/form-data">');
    res.write('<input type="file"
name="filetoupload"><br>');
    res.write('<input type="submit">');
    res.write('</form>');
    return res.end();
  }
}).listen(8080);
```

الخطوة 3: احفظ الملف

عندما يتم تحميل الملف بنجاح إلى الخادم، يتم وضعه في مجلد مؤقت.

يمكن العثور على المسار إلى هذا الدليل في كائن "الملفات"، وظيفة رد الاتصال parse() الذي تم تمريره كوسيطة ثالثة في الخاصة بالأسلوب.

لنقل الملف إلى المجلد الذي تختاره، استخدم وحدة نظام الملفات، وأعد تسمية الملف:

مثال

قم بتضمين الوحدة النمطية fs، وانقل الملف إلى المجلد الحالي:

```
var http = require('http');
var formidable = require('formidable');
var fs = require('fs');

http.createServer(function (req, res) {
  if (req.url == '/Hosini_file_upload') {
    var form
= new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      var oldpath = files.filetoupload.filepath;
      var newpath = 'C:/Users/Your Name/' +
files.filetoupload.originalFilename;
      fs.rename(oldpath,
newpath, function (err) {
        if (err) throw err;
        res.write('File uploaded and moved!');
        res.end();
```

```
      });
  });
   } else {
     res.writeHead(200, {'Content-
Type': 'text/html'});
     res.write('<form
action="Hosini_file_upload" method="post"
enctype="multipart/form-data">');
     res.write('<input type="file"
name="filetoupload"><br>');
     res.write('<input type="submit">');
     res.write('</form>');
     return res.end();
   }
}).listen(8080);
```

إرسال بريد إلكتروني

وحدة Nodemailer

تعمل وحدة **Nodemailer** على تسهيل إرسال رسائل البريد الإلكتروني من جهاز الكمبيوتر.

يمكن تنزيل وحدة **Nodemailer** وتثبيتها باستخدام **npm**:

```
C:\Users\Your Name>npm install nodemailer
```

بعد تنزيل وحدة **Nodemailer**، يمكنك تضمين الوحدة في أي تطبيق:

```
var nodemailer = require('nodemailer');
```

أرسل بريدا إلكترونيا

أنت الآن جاهز لإرسال رسائل البريد الإلكتروني من الخادم.

استخدم اسم المستخدم وكلمة المرور من مزود البريد الإلكتروني الذي حددته لإرسال بريد إلكتروني. سيوضح لك هذا الكتاب كيفية استخدام حساب **Gmail** لإرسال بريد إلكتروني:

مثال

```
var nodemailer = require('nodemailer');

var transporter
= nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'Abo_Habib_Mail@gmail.com',
    pass: 'password_Abo_Habib'
  }
});

var mailOptions = {
  from: 'Abo_Habib_Mail@gmail.com',
  to: 'Hosini_friend@yahoo.com',
  subject: 'Sending Email  with node js'
  text: 'That was easy!'
};

transporter.sendMail(mailOptions, function(
error, info){
  if (error) {
```

```
    console.log(error);
  } else {
    console.log('Email sent: ' + info.response);
  }
});
```

وهذا كل شيء! الآن خادمك قادر على إرسال رسائل البريد الإلكتروني.

ارسال و استقبال متعدد

لإرسال بريد إلكتروني إلى أكثر من مستلم واحد، قم بإضافتهم مفصولين بفواصل، إلى الخاصية "to" لكائن mailOptions:

مثال

إرسال البريد الإلكتروني إلى أكثر من عنوان:

```
var mailOptions = {
  from: 'Abo_Habib_Mail@gmail.com',
```

```
  to: 'Hosini_friend@yahoo.com, Hosini_other
friend@yahoo.com',
  subject: 'Sending Email usin ‘
  text: 'That was easy!'
}
```

إرسال HTML

في بريدك الإلكتروني، استخدم **HTML** لإرسال نص بتنسيق **HTML** بدلاً من خاصية **"text"** خاصية **"html"**:

مثال

إرسال بريد إلكتروني يحتوي على **HTML**:

```
var mailOptions = {
  from: 'Abo_Habib_Mail@gmail.com',
  to: 'Hosini_friend@yahoo.com',
  subject: 'Sending Email using ‘
  html: '<h1>Welcome</h1><p>That was
```

```
easy!</p>'
}
```

استخدام MySQL

يمكن استخدام نود جى اس في تطبيقات قواعد البيانات.

SQL. واحدة من قواعد البيانات الأكثر شعبية هي

قاعدة بيانات ماي إس كيو إل

لتتمكن من تجربة أمثلة التعليمات البرمجية، يجب أن يكون لديك SQL نسخة على جهاز الكمبيوتر مثبتًا.

يمكنك تنزيل النسخة SQL مجانية على

https://www.SQL.com/downloads/

قم بتثبيت برنامج تشغيل SQL

على جهاز الكمبيوتر، يمكنك الوصول إليه **SQL** بمجرد تشغيل باستخدام نود جى اس .

باستخدام نود جى اس ، تحتاج **SQL** للوصول إلى قاعدة بيانات **"SQL"** سيستخدم هذا الكتاب وحدة **SQL.** إلى برنامج تشغيل **NPM.** التي تم تنزيلها من

افتح محطة الأوامر وقم بتنفيذ ما ، **"SQL"** لتنزيل وتثبيت وحدة يلي:

```
C:\Users\Your Name>npm install SQL
```

لقد قمت الآن بتنزيل وتثبيت برنامج تشغيل قاعدة بيانات **SQL.**

يمكن لـ نود جى اس استخدام هذه الوحدة لمعالجة قاعدة بيانات **SQL:**

```
var SQL = require('SQL');
```

إنشاء اتصال

ابدأ بإنشاء اتصال بقاعدة البيانات.

استخدم اسم المستخدم وكلمة المرور من قاعدة بيانات SQL .

Abo_Habib_db_connection.js

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_connection.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_connection.js"

```
C:\Users\Your Name>node Abo_Habib_db_connection.js
```

والتي سوف تعطيك هذه النتيجة:

```
Connected
```

يمكنك الآن البدء في الاستعلام عن قاعدة البيانات باستخدام عبارات SQL.

الاستعلام عن قاعدة بيانات

استخدم عبارات SQL للقراءة من (أو الكتابة إلى) قاعدة بيانات SQL. ويسمى هذا أيضًا "للاستعلام" عن قاعدة البيانات.

يحتوي كائن الاتصال الذي تم إنشاؤه في المثال أعلاه على طريقة للاستعلام عن قاعدة البيانات:

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

تأخذ طريقة الاستعلام عبارات SQL كمعلمة وترجع النتيجة.

تعرف على كيفية قراءة قاعدة البيانات وكتابتها وحذفها وتحديثها في الفصول التالية.

إنشاء قاعدة بيانات

استخدم عبارة "إنشاء قاعدة بيانات" في SQL لإنشاء قاعدة بيانات:

مثال

قم بإنشاء قاعدة بيانات باسم "database_Abo_Habib":

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib"
});
```

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(

"CREATE DATABASEdatabase_Abo_Habib",

function (err, result) {
    if (err) throw err;
    console.log("Database created");
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_create_db.js" وقم بتشغيل الملف:

تشغيل "Abo_Habib_create_db.js"

```
C:\Users\Your Name>node
Abo_Habib_create_db.js
```

والتي سوف تعطيك هذه النتيجة:

```
Connected!
Database created
```

إنشاء جدول

لإنشاء جدول في SQL، استخدم عبارة "CREATE TABLE".

تأكد من تحديد اسم قاعدة البيانات عند إنشاء الاتصال:

مثال

قم بإنشاء جدول باسم "العملاء":

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE customers (name
VARCHAR(255), address VARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_create_table.js" وقم بتشغيل الملف

"Abo_Habib_create_table.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_create_table.js
```

والتي سوف تعطيك هذه النتيجة

```
Connected!
Table created
```

المفتاح الأساسي

عند إنشاء جدول، يجب عليك أيضًا إنشاء عمود يحتوي على مفتاح فريد لكل سجل.

يمكن القيام بذلك عن طريق تعريف عمود باسم "INT AUTO_INCREMENT PRIMARY KEY" والذي سيقوم

بإدراج رقم فريد لكل سجل. ابتداءً من 1، وزيادة بمقدار واحد لكل سجل.

مثال

قم بإنشاء مفتاح أساسي عند إنشاء الجدول:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE customers (id INT
AUTO_INCREMENT PRIMARY KEY, name
VARCHAR(255), address VARCHAR(255))";
```

```
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});
```

إذا كان الجدول موجودًا بالفعل، فاستخدم الكلمة الأساسية **ALTER TABLE:**

مثال

إنشاء مفتاح أساسي في جدول موجود:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
```

```
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "ALTER TABLE customers ADD
COLUMN id INT AUTO_INCREMENT PRIMARY
KEY";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table altered");
  });
});
```

إدراج في الجدول

استخدم عبارة "INSERT INTO". لملء جدول في SQL.

مثال

أدخل سجلاً في جدول "العملاء":

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name,
address) VALUES ('Company Inc', 'Giza 37')";
```

```
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_insert.js"، وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_insert.js"

```
C:\Users\Your Name>node
Abo_Habib_db_insert.js
```

والتي سوف تعطيك هذه النتيجة:

```
Connected!
1 record inserted
```

إدراج سجلات متعددة

لإدراج أكثر من سجل واحد، أنشئ مصفوفة تحتوي على القيم، والتي سيتم استبدالها بمصفوفة ،SQL وأدخل علامة استفهام في القيم:

```
INSERT INTO customers (name, address)
VALUES ?
```

مثال

املأ جدول "العملاء" بالبيانات:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});
```

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name,
address) VALUES ?";
  var values = [
    ['Amro', 'Giza 71'],
    ['HAbib', 'Al_Badrashen 4'],
    ['Mohamed', 'Apple st 652'],
    ['Ahmed', 'ElSudeya 21'],
    ['Abo_Ali', 'Omar  345'],
    ['Sandy', 'Ocean blvd 2'],
    ['Abo_saly', 'Green Grass 1'],
    ['Fatma', 'Sky st 331'],
    ['Susan', 'One way 98'],
    ['Sameer', 'Yellow Garden 2'],
    ['Ben', 'Park Lane 38'],
    ['Osman', 'Central st 954'],
    ['ghaled', 'Main Road 989'],
    ['Mahmoud', 'Sideway 1633']
  ];
  con.query(sql, [values], function (err,
result) {
```

```
    if (err) throw err;
    console.log("Number of records inserted:
" + result.affectedRows);
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_insert_multple.js"، وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_insert_multiple.js"

```
C:\Users\Your Name>node
Abo_Habib_db_insert_multiple.js
```

والتي سوف تعطيك هذه النتيجة:

```
Connected!
Number of records inserted: 14
```

كائن النتيجة

عند تنفيذ استعلام، يتم إرجاع كائن النتيجة.

يحتوي كائن النتيجة على معلومات حول كيفية تأثير الاستعلام على الجدول.

يبدو الكائن الناتج الذي تم إرجاعه من المثال أعلاه كما يلي:

```
{
  fieldCount: 0,
  affectedRows: 14,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '\'Records:14  Duplicated: 0
Warnings: 0',
  protocol41: true,
  changedRows: 0
}
```

يمكن عرض قيم الخصائص على النحو التالي:

مثال

إرجاع عدد الصفوف المتأثرة:

```
console.log(result.affectedRows)
```

والتي سوف تنتج هذه النتيجة:

```
14
```

الحصول على المعرف

بالنسبة للجداول التي تحتوي على حقل معرف الزيادة التلقائية، يمكنك الحصول على معرف الصف الذي أدرجته للتو عن طريق سؤال الكائن الناتج.

ملاحظة: لتتمكن من الحصول على المعرف المدرج، يمكن إدراج صف واحد فقط .

مثال

قم بإدراج سجل في جدول "العملاء"، ثم قم بإرجاع المعرف:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "INSERT INTO customers (name,
address) VALUES ('ABo Habib', 'Blue Village
1')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted, ID:
```

```
  " + result.insertId);
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_insert_id.js"، وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_insert_id.js"

```
C:\Users\Your Name>node
Abo_Habib_db_insert_id.js
```

والذي سيعطيك شيئًا مثل هذا في المقابل:

```
1 record inserted, ID: 15
```


تابع SQL

الاختيار من الجدول

استخدم عبارة SQL، لتحديد بيانات من جدول في "SELECT".

مثال

حدد كافة السجلات من جدول "العملاء"، واعرض الكائن الناتج:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM
customers", function (err, result, fields) {
    if (err) throw err;
```

```
    console.log(result);
  });
});
```

سيُرجع كافة الأعمدة SELECT *

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_select.js" وقم بتشغيل الملف:

"Abo_Habib_db_select.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_db_select.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { id: 1, name: 'Amro', address: 'Giza 71'},
  { id: 2, name: 'HAbib', address: 'Al_Badrashen 4'},
  { id: 3, name: 'Mohamed', address: 'Apple st 652'},
```

```
  { id: 4, name: 'Ahmed', address: 'ElSudeya 21'},
  { id: 5, name: 'Abo_Ali', address: 'Omar  345'},
  { id: 6, name: 'Sandy', address: 'Ocean blvd 2'},
  { id: 7, name: 'Abo_saly', address: 'Green Grass 1'},
  { id: 8, name: 'Fatma', address: 'Sky st 331'},
  { id: 9, name: 'Susan', address: 'One way 98'},
  { id: 10, name: 'Sameer', address: 'Yellow Garden 2'},
  { id: 11, name: 'Ben', address: 'Park Lane 38'},
  { id: 12, name: 'Osman', address: 'Central st 954'},
  { id: 13, name: 'ghaled', address: 'Main Road 989'},
  { id: 14, name: 'Mahmoud', address: 'Sideway 1633'}
]
```

اختيار الأعمدة

لتحديد بعض الأعمدة فقط في الجدول، استخدم عبارة "SELECT" متبوعة باسم العمود.

مثال

حدد الاسم والعنوان من جدول "العملاء"، واعرض كائن الإرجاع:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT name, address FROM
customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
```


احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_select2.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_select2.js"

```
C:\Users\Your Name>node
Abo_Habib_db_select2.js
```

والتي سوف تعطيك هذه النتيجة

```
[
  { name: 'Amro', address: 'Giza 71'},
  { name: 'HAbib', address: 'Al_Badrashen 4'},
  { name: 'Mohamed', address: 'Apple st 652'},
  { name: 'Ahmed', address: 'ElSudeya 21'},
  { name: 'Abo_Ali', address: 'Omar  345'},
  { name: 'Sandy', address: 'Ocean blvd 2'},
  { name: 'Abo_saly', address: 'Green Grass 1'},
  { name: 'Fatma', address: 'Sky st 331'},
  { name: 'Susan', address: 'One way 98'},
  { name: 'Sameer', address: 'Yellow Garden 2'},
  { name: 'Ben', address: 'Park Lane 38'},
  { name: 'Osman', address: 'Central st 954'},
  { name: 'ghaled', address: 'Main Road 989'},
```

```
  { name: 'Mahmoud', address: 'Sideway 1633'}
]
```

كائن النتيجة

كما ترون من نتيجة المثال أعلاه، فإن الكائن الناتج هو مصفوفة تحتوي على كل صف ككائن.

لإرجاع عنوان السجل الثالث على سبيل المثال، ما عليك سوى الرجوع إلى خاصية عنوان كائن المصفوفة الثالثة:

مثال

إرجاع عنوان السجل الثالث:

```
console.log(result[2].address);
```

والتي سوف تنتج هذه النتيجة:

```
Apple st 652
```

كائن الحقول

المعلمة الثالثة لوظيفة رد الاتصال هي مصفوفة تحتوي على معلومات حول كل حقل في النتيجة.

مثال

حدد كافة السجلات من جدول "العملاء"، واعرض كائن الحقول:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
```

```
  con.query("SELECT name, address FROM
 customers", function (err, result, fields) {
    if (err) throw err;
    console.log(fields);
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_select_fields.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_select_fields.js"

```
C:\Users\Your Name>node
Abo_Habib_db_select_fields.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
 {
  catalog: 'def',
```

```
  db: 'database_Abo_Habib',
  table: 'customers',
  orgTable: 'customers',
  name: 'name',
  orgName: 'name',
  charsetNr: 33,
  length: 765,
  type: 253,
  flags: 0,
  decimals: 0,
  default: undefined,
  zeroFill: false,
  protocol41: true
},
{
  catalog: 'def',
  db: 'database_Abo_Habib',
  table: 'customers',
  orgTable: 'customers',
  name: 'address',
  orgName: 'address',
```

```
    charsetNr: 33,
    length: 765,
    type: 253,
    flags: 0,
    decimals: 0,
    default: undefined,
    zeroFill: false,
    protocol41: true
  }
]
```

كما ترون من نتيجة المثال أعلاه، كائن الحقول عبارة عن مصفوفة تحتوي على معلومات حول كل حقل ككائن.

لإرجاع اسم الحقل الثاني على سبيل المثال، ما عليك سوى الرجوع إلى خاصية اسم عنصر المصفوفة الثانية:

مثال

إرجاع اسم الحقل الثاني:

```
console.log(fields[1].name);
```

والتي سوف تنتج هذه النتيجة:

```
address
```

البحث SQL

حدد مع مرشح

عند تحديد السجلات من جدول، يمكنك تصفية التحديد باستخدام عبارة "WHERE":

مثال

حدد السجل (السجلات) ذات العنوان "Park Lane 38":

```
var SQL = require('SQL');

var con = SQL.createConnection({
```

```
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM
customers WHERE address ='Park Lane
38'", function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```


احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_where.js" وقم بتشغيل الملف:

"Abo_Habib_db_where.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_db_where.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { id: 11, name: 'Ben', address: 'Park Lane 38'}
]
```

أحرف البدل

يمكنك أيضًا تحديد السجلات التي تبدأ أو تتضمن أو تنتهي بحرف أو عبارة معينة.

استخدم حرف البدل '%' لتمثيل صفر أو حرف واحد أو عدة أحرف:

مثال

حدد السجلات التي يبدأ عنوانها بالحرف "S":

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers
WHERE address LIKE 'S%'", function (err,
result) {
    if (err) throw err;
    console.log(result);
  });
});
```


احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_where_s.js" وقم بتشغيل الملف:

تشغيل "Abo_Habib_db_where_s.js"

```
C:\Users\Your Name>node
Abo_Habib_db_where_s.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { id: 8, name: 'Fatma', address: 'Sky st 331'},
  { id: 14, name: 'Mahmoud', address: 'Sideway
1633'}
]
```

الاثتثناءات من قيم الاستعلام

عندما تكون قيم الاستعلام متغيرات مقدمة من قبل المستخدم، يجب عليك الاثتثناءات من القيم.

وهو أسلوب شائع لاختراق الويب لتدمير SQL، وذلك لمنع حقن قاعدة البيانات أو إساءة استخدامها.

على طرق للهروب من قيم الاستعلام SQL تحتوي وحدة:

مثال

الاثتثناءات من قيم الاستعلام
الطريقة SQL.escape() باستخدام:

```
var adr = 'ElSudeya 21';
var sql = 'SELECT * FROM customers WHERE
address = ' + SQL.escape(adr);
con.query(sql, function (err, result) {
  if (err) throw err;
  console.log(result);
});
```

يمكنك أيضًا استخدام ?كعنصر نائب للقيم التي تريد
الاثتثناءات منها.

في هذه الحالة، يتم إرسال المتغير كمعلمة ثانية في طريقة
الاستعلام :

مثال

تخطي قيم الاستعلام باستخدام ? أسلوب العنصر النائب:

```
var adr = 'ElSudeya 21';
var sql = 'SELECT * FROM customers WHERE
address = ?';
con.query(sql, [adr], function (err, result) {
  if (err) throw err;
  console.log(result);
});
```

إذا كان لديك عدة عناصر نائبة، فإن المصفوفة تحتوي على قيم متعددة، بهذا الترتيب:

مثال

عناصر نائبة متعددة:

```
var name = 'Mohamed';
var adr = 'ElSudeya 21';
var sql = 'SELECT * FROM customers WHERE
name = ? OR address = ?';
con.query(sql, [name, adr], function (err,
result) {
  if (err) throw err;
  console.log(result);
});
```


فرز النتيجة

لفرز النتيجة بترتيب تصاعدي أو تنازلي استخدم عبارة ORDER BY.

بفرز النتيجة تصاعديًا ORDER BY تقوم الكلمة الأساسية بشكل افتراضي. لفرز النتيجة بترتيب تنازلي، استخدم الكلمة الأساسية DESC.

مثال

ترتيب النتيجة أبجديا حسب الاسم:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM
customers ORDER BY name", function (err,
```

```
result) {
    if (err) throw err;
    console.log(result);
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_orderby.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_orderby.js"

```
C:\Users\Your Name>node
Abo_Habib_db_orderby.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { id: 3, name: 'Mohamed', address: 'Apple st
652'},
  { id: 11, name: 'Ben', address: 'Park Lane
```

```
38'},
  { id: 7, name: 'Abo_saly', address: 'Green
Grass 1'},
  { id: 13, name: 'ghaled', address: 'Main Road
989'},
  { id: 4, name: 'Ahmed', address: 'ElSudeya
21'},
  { id: 1, name: 'Amro', address: 'Higheay 71'},
  { id: 5, name: 'Abo_Ali', address: 'Omar
345'},
  { id: 2, name: 'HAbib', address:
'Al_Badrashen 4'},
  { id: 8, name: 'Fatma', address: 'Sky st 331'},
  { id: 6, name: 'Sandy', address: 'Ocean blvd
2'},
  { id: 9, name: 'Susan', address: 'One way 98'},
  { id: 10, name: 'Sameer', address: 'Yellow
Garden 2'},
  { id: 14, name: 'Mahmoud', address: 'Sideway
1633'},
  { id: 12, name: 'Osman', address: 'Central st
954'}
]
```

الطلب حسب الوصف

لفرز النتيجة بترتيب تنازلي DESC استخدم الكلمة الأساسية.

مثال

ترتيب النتيجة عكسيا أبجديا حسب الاسم:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM
```

```
customers ORDER BY name
DESC", function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_orderby_desc.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_orderby_desc.js"

```
C:\Users\Your Name>node
Abo_Habib_db_orderby_desc.js
```

والتي سوف تعطيك هذه النتيجة

```
[
  { id: 12, name: 'Osman', address: 'Central st
```

```
954'},
  { id: 14, name: 'Mahmoud', address: 'Sideway
1633'},
  { id: 10, name: 'Sameer', address: 'Yellow
Garden 2'},
  { id: 9, name: 'Susan', address: 'One way 98'},
  { id: 6, name: 'Sandy', address: 'Ocean blvd
2'},
  { id: 8, name: 'Fatma', address: 'Sky st 331'},
  { id: 2, name: 'HAbib', address:
'Al_Badrashen 4'},
  { id: 5, name: 'Abo_Ali', address: 'Omar
345'},
  { id: 1, name: 'Amro', address: 'Higheay 71'},
  { id: 4, name: 'Ahmed', address: 'ElSudeya
21'},
  { id: 13, name: 'ghaled', address: 'Main Road
989'},
  { id: 7, name: 'Abo_saly', address: 'Green
Grass 1'},
  { id: 11, name: 'Ben', address: 'Park Lane
38'},
  { id: 3, name: 'Mohamed', address: 'Apple st
```

```
652'}
]
```

حذف جدول

يمكنك حذف جدول موجود باستخدام عبارة "DROP TABLE":

مثال

حذف جدول "العملاء":

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
```

```
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "DROP TABLE customers";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table deleted");
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_drop_table.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_drop_table.js"

```
C:\Users\Your Name>node
Abo_Habib_db_drop_table.js
```

والتي سوف تعطيك هذه النتيجة:

```
Table deleted
```

حذف فقط إذا كان موجودا

إذا كان الجدول الذي تريد حذفه محذوفًا بالفعل، أو غير موجود لأي سبب آخر، فيمكنك استخدام الكلمة الأساسية **IF EXISTS** لتجنب الحصول على خطأ.

مثال

احذف جدول "العملاء" إن وجد:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
```

```
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "DROP TABLE IF EXISTS
customers";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_drop_table_if.js" وقم بتشغيل الملف

قم بتشغيل "Abo_Habib_db_drop_table_if.js"

```
C:\Users\Your Name>node
Abo_Habib_db_drop_table_if.js
```

إذا كان الجدول موجودًا، فسيبدو الكائن الناتج كما يلي:

```
{
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverstatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

إذا كان الجدول غير موجود، فسيبدو الكائن الناتج كما يلي:

```
{
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverstatus: 2,
  warningCount: 1,
  message: '',
  protocol41: true,
```

```
  changedRows: 0
}
```

كما ترون، فإن الاختلاف الوحيد هو أن الخاصية **WaringCount** تم تعيينها على **1** إذا كان الجدول غير موجود.

تحديث الجدول

يمكنك تحديث السجلات الموجودة في الجدول باستخدام عبارة **"UPDATE":**

مثال

استبدل عمود العنوان من **"Omar 345"** إلى **"Hosam 123":**

```
var SQL = require('SQL');

var con = SQL.createConnection({
```

```
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "UPDATE customers SET address =
'Hosam 123' WHERE address = 'Omar  345'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result.affectedRows + "
record(s) updated");
  });
});
```


تحدد جملة UPDATE: في بناء جملة WHERE لاحظ جملة WHERE السجل أو السجلات التي يجب تحديثها. إذا قمت بحذف جملة WHERE، فسيتم تحديث جميع السجلات!

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_update.js" وقم بتشغيل الملف:

"Abo_Habib_db_update.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_db_update.js
```

والتي سوف تعطيك هذه النتيجة:

```
1 record(s) updated
```

كائن النتيجة

عند تنفيذ استعلام، يتم إرجاع كائن النتيجة.

يحتوي كائن النتيجة على معلومات حول كيفية تأثير الاستعلام على الجدول.

يبدو الكائن الناتج الذي تم إرجاعه من المثال أعلاه كما يلي:

```
{
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '(Rows matched: 1 Changed: 1
Warnings: 0',
  protocol41: true,
  changedRows: 1
}
```

يمكن عرض قيم الخصائص على النحو التالي:

مثال

إرجاع عدد الصفوف المتأثرة:

```
console.log(result.affectedRows)
```

والتي سوف تنتج هذه النتيجة:

1

الحد من النتيجة

يمكنك تحديد عدد السجلات التي يتم إرجاعها من الاستعلام، باستخدام عبارة "LIMIT":

مثال

حدد السجلات الخمسة الأولى في جدول "العملاء":

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});
```

```
con.connect(function(err) {
  if (err) throw err;
  var sql = "SELECT * FROM customers LIMIT
5";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_limit.js" وقم بتشغيل الملف:

تشغيل "Abo_Habib_db_limit.js"

```
C:\Users\Your Name>node
Abo_Habib_db_limit.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { id: 1, name: 'Amro', address: 'Giza 71'},
  { id: 2, name: 'HAbib', address:
'Al_Badrashen 4'},
  { id: 3, name: 'Mohamed', address: 'Apple st
652'},
  { id: 4, name: 'Ahmed', address: 'ElSudeya
21'},
  { id: 5, name: 'Abo_Ali', address: 'Omar  345'}
]
```

ابدأ البحث من موضع آخر

إذا كنت تريد إرجاع خمسة سجلات، بدءًا من السجل الثالث، فيمكنك استخدام الكلمة الأساسية **"OFFSET":**

مثال

ابدأ من الموضع 3، وقم بإرجاع السجلات الخمسة التالية:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "SELECT * FROM customers LIMIT
5 OFFSET 2";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```


ملحوظة: "الإزاحة 2" تعني البدء من المركز الثالث وليس الثاني!

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_db_offset.js" وقم بتشغيل الملف:

تشغيل "Abo_Habib_db_offset.js"

```
C:\Users\Your Name>node
Abo_Habib_db_offset.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { id: 3, name: 'Mohamed', address: 'Apple st
652'},
  { id: 4, name: 'Ahmed', address: 'ElSudeya
21'},
  { id: 5, name: 'Abo_Ali', address: 'Omar
345'},
  { id: 6, name: 'Sandy', address: 'Ocean blvd
2'},
  { id: 7, name: 'Abo_saly', address: 'Green
Grass 1'}
]
```

بناء جملة أقصر

والتي تُرجع "LIMIT 2, 5" مثل SQL يمكنك أيضًا كتابة عبارة نفس مثال الإزاحة أعلاه:

مثال

ابدأ من الموضع 3، وقم بإرجاع السجلات الخمسة التالية:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
```

```
  var sql = "SELECT * FROM customers LIMIT
2, 5";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

ملحوظة: الأرقام معكوسة: "LIMIT 2, 5" هو نفس "LIMIT 5 OFFSET 2"

ضم جدولين أو أكثر

يمكنك دمج صفوف من جدولين أو أكثر، استنادًا إلى عمود مرتبط بينهما، باستخدام عبارة JOIN.

ضع في اعتبارك أن لديك جدول "المستخدمين" وجدول "المنتجات":

المستخدمين

```
[
  { id: 1, name: 'Amro', favorite_product: 154},
  { id: 2, name: 'HAbib', favorite_product:
154},
  { id: 3, name: 'Mohamed', favorite_product:
155},
  { id: 4, name: 'Ahmed', favorite_product:},
  { id: 5, name: 'Abo_Ali', favorite_product:}
]
```

منتجات

```
[
  { id: 154, name: 'Ebraheem ' },
  { id: 155, name: 'Hamza' },
  { id: 156, name: 'Yousef }
]
```

حقل favorite_product يمكن دمج هذين الجدولين باستخدام id المستخدمين وحقل المنتجات.

مثال

حدد السجلات التي لها تطابق في كلا الجدولين:

```
var SQL = require('SQL');

var con = SQL.createConnection({
  host: "localhost",
  user: "Abo_Habib_Al_Hosiny",
  password: "password_Abo_Habib",
  database: "database_Abo_Habib"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "SELECT users.name AS user,
products.name AS favorite FROM users JOIN
products ON users.favorite_product =
products.id";
```

```
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

ملاحظة: يمكنك استخدام INNER JOIN بدلاً من JOIN. كلاهما سيعطيك نفس النتيجة.

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_db_join.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_db_join.js"

```
C:\Users\Your Name>node Abo_Habib_db_join.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { user: 'Amro', favorite: 'Ebraheem ' },
  { user: 'HAbib', favorite: 'Ebraheem ' },
  { user: 'Mohamed', favorite: 'Hamza' }
]
```

كما ترون من النتيجة أعلاه، يتم إرجاع السجلات المتطابقة في كلا الجدولين فقط.

الانضمام الأيسر

إذا كنت تريد إرجاع جميع المستخدمين، بغض النظر عما إذا كان لديهم منتج مفضل أم لا، فاستخدم عبارة **LEFT JOIN:**

مثال

حدد كافة المستخدمين والمنتج المفضل لديهم:

```
SELECT users.name AS user,
products.name AS favorite
```

```
FROM users
LEFT JOIN products ON
users.favorite_product = products.id
```

والتي سوف تعطيك هذه النتيجة

```
[
  { user: 'Amro', favorite: 'Ebraheem ' },
  { user: 'HAbib', favorite: 'Ebraheem ' },
  { user: 'Mohamed', favorite: 'Hamza' },
  { user: 'Ahmed', favorite: null },
  { user: 'Abo_Ali', favorite: null }
]
```

الإنضمام الصحيح

إذا كنت تريد إرجاع جميع المنتجات والمستخدمين الذين جعلوها مفضلة لديهم، حتى لو لم يكن هناك أي مستخدم يجعلها المفضلة لديهم، فاستخدم عبارة **RIGHT JOIN:**

مثال

حدد جميع المنتجات والمستخدم الذي جعلها مفضلة لديه:

```
SELECT users.name AS user,
products.name AS favorite
FROM users
RIGHT JOIN products ON
users.favorite_product = products.id
```

والتي سوف تعطيك هذه النتيجة

```
[
  { user: 'Amro', favorite: 'Ebraheem ' },
  { user: 'HAbib', favorite: 'Ebraheem ' },
  { user: 'Mohamed', favorite: 'Hamza' },
  { user: null, favorite: 'Yousef }
]
```

ملحوظة: هانا ومايكل، اللذان ليس لديهما منتج مفضل، لم يتم تضمينهما في النتيجة.

استخدام MongoDB

يمكن استخدام نود جى اس في تطبيقات قواعد البيانات.

قواعد بيانات MongoDB

لتتمكن من تجربة أمثلة التعليمات البرمجية، ستحتاج إلى الوصول إلى قاعدة بيانات MongoDB.

يمكنك تنزيل قاعدة بيانات MongoDB المجانية على https://www.mongodb.com .

أو ابدأ على الفور باستخدام خدمة MongoDB السحابية على https://www.mongodb.com/cloud/atlas .

MongoDB تثبيت برنامج تشغيل

باستخدام **MongoDB** دعونا نحاول الوصول إلى قاعدة بيانات نود جى اس .

الرسمي، افتح **MongoDB** لتنزيل وتثبيت برنامج تشغيل **Command Terminal** وقم بتنفيذ ما يلي:

تنزيل وتثبيت حزمة **mongodb:**

```
C:\Users\Your Name>npm install mongodb
```

لقد قمت الآن بتنزيل وتثبيت برنامج تشغيل قاعدة بيانات **mongodb.**

يمكن لـ نود جى اس استخدام هذه الوحدة لمعالجة قواعد بيانات **MongoDB:**

```
var mongo = require('mongodb');
```

MongoDB إنشاء قاعدة بيانات

إنشاء قاعدة البيانات

ابدأ بإنشاء كائن MongoDB. لإنشاء قاعدة بيانات في MongoClient. ثم حدد عنوان URL للاتصال بعنوان IP الصحيح واسم قاعدة البيانات التي تريد إنشاءها.

بإنشاء قاعدة البيانات إذا لم تكن موجودة، MongoDB سيقوم وإجراء اتصال بها.

مثال

قم بإنشاء قاعدة بيانات تسمى "database_Abo_Habib":

```
var MongoClient =
require('mongodb').MongoClient;
var url
= "mongodb://localhost:27017/database_Ab
o_Habib";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_create_mongo_db.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_create_mongo_db.js"

```
C:\Users\Your Name>node
Abo_Habib_create_mongo_db.js
```

والتي سوف تعطيك هذه النتيجة

```
Database created!
```

لا يتم إنشاء قاعدة بيانات حتى تحصل MongoDB. هام: في على المحتوى!

حتى تقوم بإنشاء مجموعة (جدول)، مع MongoDB ينتظر مستند واحد على الأقل (سجل) قبل أن يقوم فعليًا بإنشاء قاعدة البيانات (والمجموعة).

SQL هي نفس الجدول في MongoDB المجموعة في

إنشاء مجموعة

MongoDB. لإنشاء مجموعة في
الطريقة createCollection() استخدم:

مثال

قم بإنشاء مجموعة تسمى "العملاء":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");

dbo.createCollection("customers", function(
err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_createcollection.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_createcollection.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_createcollection.js
```

والتي سوف تعطيك هذه النتيجة:

```
Collection created!
```

هام: في MongoDB، لا يتم إنشاء المجموعة حتى تحصل على المحتوى!

MongoDB ينتظر حتى تقوم بإدراج مستند قبل أن يقوم بإنشاء المجموعة فعليًا.

إدراج في المجموعة

في MongoDB، لإدراج سجل، أو مستند كما يطلق عليه في مجموعة، نستخدم الطريقة insertOne().

SQL هو نفس السجل في MongoDB المستند في

هي كائن يحتوي على insertOne() المعلمة الأولى للطريقة الاسم (الأسماء) والقيمة (القيم) لكل حقل في المستند الذي تريد إدراجه.

يتطلب أيضًا وظيفة رد اتصال حيث يمكنك التعامل مع أي أخطاء أو نتيجة الإدراج:

مثال

أدخل مستندًا في مجموعة "العملاء":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var Hosini_obj = { name: "Company Inc",
address: "Giza 37" };

dbo.collection("customers").insertOne(Hosi
ni_obj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_mongodb_insert.js" وقم بتشغيل الملف:

"Abo_Habib_mongodb_insert.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_mongodb_insert.js
```

والتي سوف تعطيك هذه النتيجة:

```
1 document inserted
```

ملاحظة: إذا حاولت إدراج مستندات في مجموعة غير موجودة،
فسيقوم MongoDB بإنشاء المجموعة تلقائيًا.

إدراج مستندات متعددة

لإدراج مستندات متعددة في مجموعة في MongoDB، نستخدم هذه insertMany() الطريقة.

هي مصفوفة من insertMany() المعلمة الأولى للطريقة الكائنات تحتوي على البيانات التي تريد إدراجها.

يتطلب أيضًا وظيفة رد اتصال حيث يمكنك التعامل مع أي أخطاء أو نتيجة الإدراج:

مثال

أدخل مستندات متعددة في مجموعة "العملاء":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
```

```
 var Hosini_obj = [
   { name: 'Amro', address: 'Giza 71'},
   { name: 'HAbib', address: 'Al_Badrashen 4'},
   { name: 'Mohamed', address: 'Apple st
652'},
   { name: 'Ahmed', address: 'ElSudeya 21'},
   { name: 'Abo_Ali', address: 'Omar  345'},
   { name: 'Sandy', address: 'Ocean blvd 2'},
   { name: 'Abo_saly', address: 'Green Grass
1'},
   { name: 'Fatma', address: 'Sky st 331'},
   { name: 'Susan', address: 'One way 98'},
   { name: 'Sameer', address: 'Yellow Garden
2'},
   { name: 'Ben', address: 'Park Lane 38'},
   { name: 'Osman', address: 'Central st 954'},
   { name: 'ghaled', address: 'Main Road 989'},
   { name: 'Mahmoud', address: 'Sideway
1633'}
 ];

dbo.collection("customers").insertMany(Hos
ini_obj, function(err, res) {
```

```
    if (err) throw err;
    console.log("Number of documents
inserted: " + res.insertedCount);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_insert_multiple.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_insert_multiple.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_insert_multiple.js
```

والتي سوف تعطيك هذه النتيجة:

```
Number of documents inserted: 14
```

كائن النتيجة

الطريقة، يتم إرجاع كائن النتيجة insertMany() عند تنفيذ.

يحتوي الكائن الناتج على معلومات حول كيفية تأثير الإدراج على قاعدة البيانات.

يبدو الكائن الذي تم إرجاعه من المثال أعلاه كما يلي:

```
{
  result: { ok: 1, n: 14 },
  ops: [
    { name: 'Amro', address: 'Giza 71', _id:
58fdbf5c0ef8a50b4cdd9a84 },
    { name: 'HAbib', address: 'Al_Badrashen 4',
_id: 58fdbf5c0ef8a50b4cdd9a85 },
    { name: 'Mohamed', address: 'Apple st 652',
_id: 58fdbf5c0ef8a50b4cdd9a86 },
    { name: 'Ahmed', address: 'ElSudeya 21',
_id: 58fdbf5c0ef8a50b4cdd9a87 },
    { name: 'Abo_Ali', address: 'Omar  345', _id:
58fdbf5c0ef8a50b4cdd9a88 },
    { name: 'Sandy', address: 'Ocean blvd 2',
_id: 58fdbf5c0ef8a50b4cdd9a89 },
```

```
    { name: 'Abo_saly', address: 'Green Grass
1', _id: 58fdbf5c0ef8a50b4cdd9a8a },
    { name: 'Fatma', address: 'Sky st 331', _id:
58fdbf5c0ef8a50b4cdd9a8b },
    { name: 'Susan', address: 'One way 98', _id:
58fdbf5c0ef8a50b4cdd9a8c },
    { name: 'Sameer', address: 'Yellow Garden
2', _id: 58fdbf5c0ef8a50b4cdd9a8d },
    { name: 'Ben', address: 'Park Lane 38', _id:
58fdbf5c0ef8a50b4cdd9a8e },
    { name: 'Osman', address: 'Central st 954',
_id: 58fdbf5c0ef8a50b4cdd9a8f },
    { name: 'ghaled', address: 'Main Road 989',
_id: 58fdbf5c0ef8a50b4cdd9a90 },
    { name: 'Mahmoud', address: 'Sideway
1633', _id: 58fdbf5c0ef8a50b4cdd9a91 } ],
  insertedCount: 14,
  insertedIds: [
    58fdbf5c0ef8a50b4cdd9a84,
    58fdbf5c0ef8a50b4cdd9a85,
    58fdbf5c0ef8a50b4cdd9a86,
    58fdbf5c0ef8a50b4cdd9a87,
    58fdbf5c0ef8a50b4cdd9a88,
```

```
    58fdbf5c0ef8a50b4cdd9a89,
    58fdbf5c0ef8a50b4cdd9a8a,
    58fdbf5c0ef8a50b4cdd9a8b,
    58fdbf5c0ef8a50b4cdd9a8c,
    58fdbf5c0ef8a50b4cdd9a8d,
    58fdbf5c0ef8a50b4cdd9a8e,
    58fdbf5c0ef8a50b4cdd9a8f
    58fdbf5c0ef8a50b4cdd9a90,
    58fdbf5c0ef8a50b4cdd9a91 ]
}
```

يمكن عرض قيم الخصائص على النحو التالي:

مثال

إرجاع عدد المستندات المدرجة:

```
console.log(res.insertedCount)
```

والتي سوف تنتج هذه النتيجة:

```
14
```

الحقل _id

حقلاً لك ويعين MongoDB فسيضيف ،_id إذا لم تحدد حقلاً معرفًا فريدًا لكل مستند.

يتم تحديد أي حقل، وكما ترون من _id في المثال أعلاه لم بتعيين معرف _ فريد لكل MongoDB الكائن الناتج، قام مستند.

فيجب أن تكون القيمة فريدة لكل ،_id إذا قمت بتحديد الحقل مستند:

مثال

الحقول_id قم بإدراج ثلاثة سجلات في جدول "المنتجات"، مع المحددة:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var Hosini_obj = [
    { _id: 154, name: 'Ebraheem '},
    { _id: 155, name: 'Sameh'},
    { _id: 156, name: 'Hosaam'}
  ];

dbo.collection("products").insertMany(Hosi
ni_obj, function(err, res) {
    if (err) throw err;
    console.log(res);
    db.close();
  });
});
```


احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_insert_id.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_insert_id.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_insert_id.js
```

والتي سوف تعطيك هذه النتيجة

```
{
  result: { ok: 1, n: 3 },
  ops: [
    { _id: 154, name: 'Ebraheem  },
    { _id: 155, name: 'Sameh },
    { _id: 156, name: 'Hosaam } ],
  insertedCount: 3,
  insertedIds: [
    154,
    155,
    156 ]
}
```

للعثور findOne و find نستخدم طريقتي MongoDB في على البيانات في مجموعة.

للعثور على البيانات SELECT تمامًا مثلما يتم استخدام عبارة SQL في جدول في قاعدة بيانات.

ابحث عن ناتج واحد

يمكننا استخدام MongoDB. لتحديد البيانات من مجموعة في الطريقة findOne() هذه.

بإرجاع التواجد الأول في التحديد findOne() تقوم الطريقة.

هي كائن استعلام. في findOne() المعلمة الأولى للأسلوب هذا المثال، نستخدم كائن استعلام فارغًا، والذي يحدد جميع المستندات في المجموعة (ولكنه يقوم بإرجاع المستند الأول فقط).

مثال

ابحث عن المستند الأول في مجموعة العملاء:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");

dbo.collection("customers").findOne({}, func
tion(err, result) {
    if (err) throw err;
    console.log(result.name);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_findone.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_findone.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_findone.js
```

والتي سوف تعطيك هذه النتيجة:

```
Company Inc.
```

اوجد كل النتائج

يمكننا أيضًا MongoDB. لتحديد البيانات من جدول في الطريقة find() استخدام هذه.

تقوم الطريقة find() بإرجاع كافة التكرارات في التحديد.

هي كائن استعلام. في هذا find() المعلمة الأولى للأسلوب المثال، نستخدم كائن استعلام فارغًا، والذي يقوم بتحديد كافة المستندات الموجودة في المجموعة.

لا توجد معلمات في طريقة البحث () تمنحك نفس النتيجة مثل SELECT * في SQL.

مثال

ابحث عن جميع المستندات في مجموعة العملاء:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.collection("customers").find({}).toArra
y(function(err, result) {
```

```
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_find.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_find.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_find.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a84 , name:
'Amro', address: 'Giza 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85 , name:
```

```
'HAbib', address: 'Al_Badrashen 4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a86 , name:
'Mohamed', address: 'Apple st 652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87 , name:
'Ahmed', address: 'ElSudeya 21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88 , name:
'Abo_Ali', address: 'Omar  345'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89 , name:
'Sandy', address: 'Ocean blvd 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a , name:
'Abo_saly', address: 'Green Grass 1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b , name:
'Fatma', address: 'Sky st 331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c , name:
'Susan', address: 'One way 98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d , name:
'Sameer', address: 'Yellow Garden 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8e , name:
'Ben', address: 'Park Lane 38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8f , name:
'Osman', address: 'Central st 954'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90 , name:
'ghaled', address: 'Main Road 989'},
```

```
  { _id: 58fdbf5c0ef8a50b4cdd9a91 , name:
'Mahmoud', address: 'Sideway 1633'}
]
```

جد بعض

الكائن الذي projection هي find() المعلمة الثانية للطريقة يصف الحقول التي سيتم تضمينها في النتيجة.

هذه المعلمة اختيارية، وإذا تم حذفها، سيتم تضمين جميع الحقول في النتيجة.

مثال

قم بإرجاع الحقلين "الاسم" و"العنوان" لجميع المستندات الموجودة في مجموعة العملاء:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.collection("customers").find({}, { projec
tion: { _id: 0, name: 1,
address: 1 } }).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```


احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_find_fields.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_find_fields.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_find_fields.js
```

والتي سوف تعطيك هذه النتيجة

```
[
  { name: 'Amro', address: 'Giza 71'},
  { name: 'HAbib', address: 'Al_Badrashen 4'},
  { name: 'Mohamed', address: 'Apple st 652'},
  { name: 'Ahmed', address: 'ElSudeya 21'},
  { name: 'Abo_Ali', address: 'Omar  345'},
  { name: 'Sandy', address: 'Ocean blvd 2'},
  { name: 'Abo_saly', address: 'Green Grass
1'},
  { name: 'Fatma', address: 'Sky st 331'},
  { name: 'Susan', address: 'One way 98'},
  { name: 'Sameer', address: 'Yellow Garden
2'},
  { name: 'Ben', address: 'Park Lane 38'},
  { name: 'Osman', address: 'Central st 954'},
  { name: 'ghaled', address: 'Main Road 989'},
  { name: 'Mahmoud', address: 'Sideway
```

```
1633'}
]
```

إلا إذا) غير مسموح لك بتحديد القيمتين 0 و1 في نفس الكائن
إذا قمت بتحديد حقل بالقيمة 0، .(id_ كان أحد الحقول هو حقل
:فإن كافة الحقول الأخرى تحصل على القيمة 1، والعكس صحيح

مثال

:سيستبعد هذا المثال "العنوان" من النتيجة

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.collection("customers").find({}, { projec
tion: { address: 0 } }).toArray(function(err,
```

```
result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

يجب عليك تعيين قيمته على 0، _id لاستبعاد الحقل:

مثال

سيُرجع هذا المثال حقل "الاسم" فقط:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
```

```
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.collection("customers").find({}, { projec
tion: { _id: 0,
name: 1 } }).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

مثال

هذا المثال سيعطيك نفس نتيجة المثال الأول؛ إرجاع جميع الحقول باستثناء حقل _id:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.collection("customers").find({}, { projec
tion: { _id: 0 } }).toArray(function(err, result)
{
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```


مثال

ستحصل على خطأ إذا قمت بتحديد القيمتين 0 و1 في نفس الكائن (إلا إذا كان أحد الحقول هو حقل _id):

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.collection("customers").find({}, { projec
tion: { name: 1,
address: 0 } }).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```


كائن النتيجة

كما ترون من نتيجة المثال أعلاه، يمكن تحويل النتيجة إلى مصفوفة تحتوي على كل مستند ككائن.

لإرجاع عنوان المستند الثالث على سبيل المثال، ما عليك سوى الرجوع إلى خاصية عنوان كائن المصفوفة الثالثة:

مثال

قم بإرجاع عنوان المستند الثالث:

```
console.log(result[2].address);
```

والتي سوف تنتج هذه النتيجة:

```
Apple st 652
```

استعلام MongoDB

تصفية النتيجة

عند البحث عن مستندات في مجموعة، يمكنك تصفية النتيجة باستخدام كائن استعلام.

هي كائن استعلام، ويتم find() الوسيطة الأولى للأسلوب استخدامها للحد من البحث.

مثال

ابحث عن المستندات التي تحمل العنوان "Park Lane 38":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var query = { address: "Park Lane 38" };

dbo.collection("customers").find(query).toA
rray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_query.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_query.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_query.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a8e , name:
'Ben', address: 'Park Lane 38' }
]
```

تصفية مع التعبيرات العادية

يمكنك كتابة تعبيرات عادية للعثور على ما تبحث عنه بالضبط.

لا يمكن استخدام التعبيرات العادية إلا للاستعلام عن السلاسل النصيه .

للعثور فقط على المستندات التي يبدأ فيها حقل "العنوان" بالحرف "S"، استخدم التعبير العادي: /^S/

مثال

ابحث عن المستندات التي يبدأ عنوانها بالحرف "S":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var query = { address: /^S/ };

dbo.collection("customers").find(query).toA
rray(function(err, result) {
```

```
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_query_s.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_query_s.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_query_s.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a8b , name:
'Fatma', address: 'Sky st 331' },
```

```
  { _id: 58fdbf5c0ef8a50b4cdd9a91 , name:
'Mahmoud', address: 'Sideway 1633' }
]
```


طريقة فرز النتيجة

الطريقة لفرز النتيجة بترتيب تصاعدي أو تنازلي sort() استخدم.

معلمة واحدة، وهي كائن يحدد ترتيب الفرز sort() تأخذ الطريقة.

مثال

ترتيب النتيجة أبجديا حسب الاسم:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var Hosini_sort = { name: 1 };

dbo.collection("customers").find().sort(Hosi
ni_sort).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_sort.js"
وقم بتشغيل الملف:

تشغيل "Abo_Habib_sort.js"

```
C:\Users\Your Name>node
Abo_Habib_sort.js
```

والتي سوف تعطيك هذه النتيجة

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a86, name:
'Mohamed', address: 'Apple st 652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8e, name:
'Ben', address: 'Park Lane 38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a, name:
'Abo_saly', address: 'Green Grass 1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90, name:
'ghaled', address: 'Main Road 989'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87, name:
'Ahmed', address: 'ElSudeya 21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a84, name:
'Amro', address: 'Giza 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88, name:
'Abo_Ali', address: 'Omar  345'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85, name:
'HAbib', address: 'Al_Badrashen 4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b, name:
```

```
'Fatma', address: 'Sky st 331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89, name:
'Sandy', address: 'Ocean blvd 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c, name:
'Susan', address: 'One way 98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d, name:
'Sameer', address: 'Yellow Garden 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a91, name:
'Mahmoud', address: 'Sideway 1633'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8f, name:
'Osman', address: 'Central st 954'}
]
```

ترتيب تنازلي

استخدم القيمة **-1** في كائن الفرز للفرز تنازليًا.

```
تصاعدي // { الاسم: 1 }
تنازلي // { الاسم: -1 }
```

مثال

ترتيب النتيجة عكسيا أبجديا حسب الاسم:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var Hosini_sort = { name: -1 };

dbo.collection("customers").find().sort(Hosi
ni_sort).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_sort_desc.js" وقم بتشغيل الملف:

"Abo_Habib_sort_desc.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_sort_desc.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a8f, name:
'Osman', address: 'Central st 954'},
  { _id: 58fdbf5c0ef8a50b4cdd9a91, name:
'Mahmoud', address: 'Sideway 1633'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d, name:
'Sameer', address: 'Yellow Garden 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c, name:
'Susan', address: 'One way 98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89, name:
'Sandy', address: 'Ocean blvd 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b, name:
```

```
'Fatma', address: 'Sky st 331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85, name:
'HAbib', address: 'Al_Badrashen 4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88, name:
'Abo_Ali', address: 'Omar  345'},
  { _id: 58fdbf5c0ef8a50b4cdd9a84, name:
'Amro', address: 'Giza 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87, name:
'Ahmed', address: 'ElSudeya 21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90, name:
'ghaled', address: 'Main Road 989'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a, name:
'Abo_saly', address: 'Green Grass 1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8e, name:
'Ben', address: 'Park Lane 38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a86, name:
'Mohamed', address: 'Apple st 652'}
]
```

تشغيل "Abo_Habib_sort.js"

```
C:\Users\Your Name>node
Abo_Habib_sort.js
```

والتي سوف تعطيك هذه النتيجة

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a86, name:
'Mohamed', address: 'Apple st 652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8e, name:
'Ben', address: 'Park Lane 38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a, name:
'Abo_saly', address: 'Green Grass 1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90, name:
'ghaled', address: 'Main Road 989'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87, name:
'Ahmed', address: 'ElSudeya 21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a84, name:
'Amro', address: 'Giza 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88, name:
'Abo_Ali', address: 'Omar  345'},
```

```
  { _id: 58fdbf5c0ef8a50b4cdd9a85, name:
'HAbib', address: 'Al_Badrashen 4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b, name:
'Fatma', address: 'Sky st 331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89, name:
'Sandy', address: 'Ocean blvd 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c, name:
'Susan', address: 'One way 98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d, name:
'Sameer', address: 'Yellow Garden 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a91, name:
'Mahmoud', address: 'Sideway 1633'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8f, name:
'Osman', address: 'Central st 954'}
]
```


حذف المجموعة

يمكنك حذف جدول، أو مجموعة كما يطلق عليها في MongoDB، باستخدام هذه الطريقة drop().

دالة رد اتصال تحتوي على كائن الخطأ drop() تأخذ الطريقة ومعلمة النتيجة التي تُرجع صحيحًا إذا تم حذف المجموعة بنجاح، وإلا فإنها تُرجع خطأ.

مثال

حذف جدول "العملاء":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.collection("customers").drop(function(
err, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection
deleted");
    db.close();
```

```
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_drop.js" وقم بتشغيل الملف:

تشغيل "Abo_Habib_drop.js"

```
C:\Users\Your Name>node Abo_Habib_drop.js
```

والتي سوف تعطيك هذه النتيجة:

```
Collection deleted
```

---

db.dropCollection

الطريقة لحذف dropCollection() يمكنك أيضًا استخدام جدول (مجموعة).

معلمتين: اسم المجموعة dropCollection() تأخذ الطريقة ووظيفة رد الاتصال.

مثال

احذف مجموعة "العملاء" باستخدام dropCollection():

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  dbo.dropCollection("customers", function(e
rr, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection
deleted");
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_dropcollection.js" وقم بتشغيل الملف:

"Abo_Habib_dropcollection.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_dropcollection.js
```

والتي سوف تعطيك هذه النتيجة:

```
Collection deleted
```


تحديث MongoDB

تحديث المستند

MongoDB يمكنك تحديث سجل أو مستند كما يطلق عليه في باستخدام الطريقة updateOne().

هي كائن استعلام يحدد updateOne() المعلمة الأولى للطريقة المستند الذي سيتم تحديثه.

ملاحظة: إذا عثر الاستعلام على أكثر من سجل واحد، فسيتم تحديث التواجد الأول فقط.

المعلمة الثانية هي كائن يحدد القيم الجديدة للمستند.

مثال

قم بتحديث المستند بالعنوان "Omar 345" = إلى الاسم والعنوان "Hosam 123" = "ebarheem":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://127.0.0.1:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var Hosini_query = { address: "Omar 345" };
  var newvalues = { $set: {name: "ebarheem",
address: "Hosam 123" } };

dbo.collection("customers").updateOne(Hos
ini_query, newvalues, function(err, res) {
    if (err) throw err;
    console.log("1 document updated");
    db.close();
  });
});
```


احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_update_one.js" ثوقم بتشغيل الملف

"Abo_Habib_update_one.js" قم بتشغيل

```
C:\Users\Your Name>node
Abo_Habib_update_one.js
```

والتي سوف تعطيك هذه النتيجة

```
1 document updated
```

تحديث حقول المحددة فقط

عامل التشغيل، يتم تحديث الحقول $set عند استخدام المحددة فقط:

مثال

قم بتحديث العنوان من "Omar 345" إلى "Hosam 123":

```
...
  var Hosini_query = { address: "Omar 345" };
  var newvalues = { $set: { address: "Hosam
123" } };
```

```
dbo.collection("customers").updateOne(Hos
ini_query, newvalues, function(err, res) {
...
```

تحديث العديد من المستندات

لتحديث كافة المستندات التي تفي بمعايير الاستعلام، استخدم updateMany() الطريقة.

مثال

تحديث جميع المستندات التي يبدأ الاسم فيها بالحرف "S":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://127.0.0.1:27017/";
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
  var Hosini_query = { address: /^S/ };
  var newvalues = {$set: {name: "Minnie"} };
  dbo.collection("customers").updateMany(H
osini_query, newvalues, function(err, res) {
    if (err) throw err;
    console.log(res.result.nModified + "
document(s) updated");
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى
"Abo_Habib_update_many.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_update_many.js"

```
C:\Users\Your Name>node
Abo_Habib_update_many.js
```

والتي سوف تعطيك هذه النتيجة:

```
2 document(s) updated
```

كائن النتيجة

بإرجاع كائن updateMany() الأساليب updateOne() تقوم يحتوي على معلومات حول كيفية تأثير التنفيذ على قاعدة البيانات.

ليس من المهم فهم معظم المعلومات، ولكن هناك كائن واحد داخل الكائن يسمى "النتيجة" والذي يخبرنا ما إذا كان التنفيذ يسير على ما يرام، وعدد المستندات التي تأثرت.

يبدو الكائن الناتج كما يلي:

```
{ n: 1, nModified: 2, ok: 1 }
```

يمكنك استخدام هذا الكائن لإرجاع عدد المستندات المحدثة:

مثال

إرجاع عدد المستندات المحدثة:

```
console.log(res.result.nModified);
```

والتي سوف تنتج هذه النتيجة:

```
2
```

الحد من عدد النتائج

الطريقة limit() نستخدم MongoDB، للحد من النتيجة في.

معاملًا واحدًا، وهو رقم يحدد عدد limit() تأخذ الطريقة المستندات التي سيتم إرجاعها.

اعتبر أن لديك مجموعة "العملاء":

عملاء

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a84 , name:
'Amro', address: 'Giza 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85 , name:
'HAbib', address: 'Al_Badrashen 4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a86 , name:
'Mohamed', address: 'Apple st 652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87 , name:
'Ahmed', address: 'ElSudeya 21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88 , name:
'Abo_Ali', address: 'Omar  345'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89 , name:
'Sandy', address: 'Ocean blvd 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a , name:
'Abo_saly', address: 'Green Grass 1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b , name:
'Fatma', address: 'Sky st 331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c , name:
'Susan', address: 'One way 98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d , name:
'Sameer', address: 'Yellow Garden 2'},
```

```
  { _id: 58fdbf5c0ef8a50b4cdd9a8e , name:
'Ben', address: 'Park Lane 38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8f , name:
'Osman', address: 'Central st 954'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90 , name:
'ghaled', address: 'Main Road 989'},
  { _id: 58fdbf5c0ef8a50b4cdd9a91 , name:
'Mahmoud', address: 'Sideway 1633'}
]
```

مثال

حدد النتيجة بإرجاع 5 مستندات فقط:

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("database_Abo_Habib");
```

```
  dbo.collection("customers").find().limit(5).t
oArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_limit.js" وقم بتشغيل الملف

قم بتشغيل "Abo_Habib_mongodb_limit.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_limit.js
```

والتي سوف تعطيك هذه النتيجة

عملاء

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a84 , name:
'Amro', address: 'Giza 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85 , name:
'HAbib', address: 'Al_Badrashen 4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a86 , name:
'Mohamed', address: 'Apple st 652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87 , name:
'Ahmed', address: 'ElSudeya 21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88 , name:
'Abo_Ali', address: 'Omar  345'}
]
```

كما ترون من النتيجة أعلاه، تم إرجاع المستندات الخمسة الأولى فقط.

ضم المجموعات

ليست قاعدة بيانات علائقية، ولكن يمكنك إجراء MongoDB صلة خارجية يسرى باستخدام المرحلة $lookup.

تحديد المجموعة التي تريد ضمها $lookup تتيح لك المرحلة إلى المجموعة الحالية، وأي الحقول يجب أن تتطابق.

اعتبر أن لديك مجموعة "طلبات" ومجموعة "منتجات":

طلبات

```
[
  { _id: 1, product_id: 154, status: 1 }
]
```

منتجات

```
[
  { _id: 154, name: 'Ebraheem ' },
  { _id: 155, name: 'Hamza' },
  { _id: 156, name: 'Yousef }
]
```

مثال

انضم إلى مستندات "المنتجات" المطابقة لمجموعة "الطلبات":

```
var MongoClient =
require('mongodb').MongoClient;
var url = "mongodb://127.0.0.1:27017/";

MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("database_Abo_Habib");
 dbo.collection('orders').aggregate([
  { $lookup:
    {
     from: 'products',
     localField: 'product_id',
     foreignField: '_id',
     as: 'orderdetails'
    }
  }
```

```
    ]).toArray(function(err, res) {
    if (err) throw err;
    console.log(JSON.stringify(res));
    db.close();
  });
});
```

احفظ الكود أعلاه في ملف يسمى "Abo_Habib_mongodb_join.js" وقم بتشغيل الملف:

قم بتشغيل "Abo_Habib_mongodb_join.js"

```
C:\Users\Your Name>node
Abo_Habib_mongodb_join.js
```

والتي سوف تعطيك هذه النتيجة:

```
[
  { "_id": 1, "product_id": 154, "status": 1,
"orderdetails": [
```

```
    { "_id": 154, "name": "Ebraheem " } ]
  }
]
```

كما ترون من النتيجة أعلاه، يتم تضمين المستند المطابق من مجموعة المنتجات في مجموعة الطلبات كمصفوفة.

يحتوي نود جى اس على مجموعة من الوحدات المضمنة التي يمكنك استخدامها دون أي تثبيت إضافي.

فيما يلي قائمة بالوحدات المضمنة في الإصدار **6.10.3** من نود جى اس:

Module	وصف
assert	يوفر مجموعة من اختبارات التأكيد
buffer	للتعامل مع البيانات الثنائية
child_process	لتشغيل عملية فرعية
cluster	لتقسيم عملية عقدة واحدة إلى عمليات متعددة
crypto	للتعامل مع وظائف التشفير OpenSSL
dgram	يوفر تنفيذ مآخذ مخطط بيانات UDP
dns	ووظائف تحليل الأسماء DNS للقيام بعمليات بحث
domain	إهمال. لمعالجة الأخطاء غير المعالجة
events	للتعامل مع الأحداث

fs	للتعامل مع نظام الملفات
http	لجعل نود جى اس يعمل كخادم HTTP
https	لجعل نود جى اس يعمل كخادم HTTPS.
net	لإنشاء الخوادم والعملاء
os	يوفر معلومات حول نظام التشغيل
path	للتعامل مع مسارات الملفات
punycode	إهمال. نظام ترميز الأحرف
querystring	للتعامل مع سلاسل استعلام URL
readline	للتعامل مع التدفقات القابلة للقراءة سطرًا واحدًا في كل مرة
stream	للتعامل مع البيانات المتدفقة
string_decoder	لفك تشفير الكائنات المخزنة إلى سلاسل
timers	لتنفيذ وظيفة بعد عدد معين من الملي ثانية
tls	لتنفيذ بروتوكولات SSL وTLS
tty	يوفر الفئات المستخدمة من قبل محطة النص
url	لتحليل سلاسل URL
util	للوصول إلى وظائف المرافق
v8	للوصول إلى معلومات حول V8 (محرك JavaScript)
vm	في جهاز افتراضي JavaScript لتجميع كود
zlib	لضغط أو فك ضغط الملفات