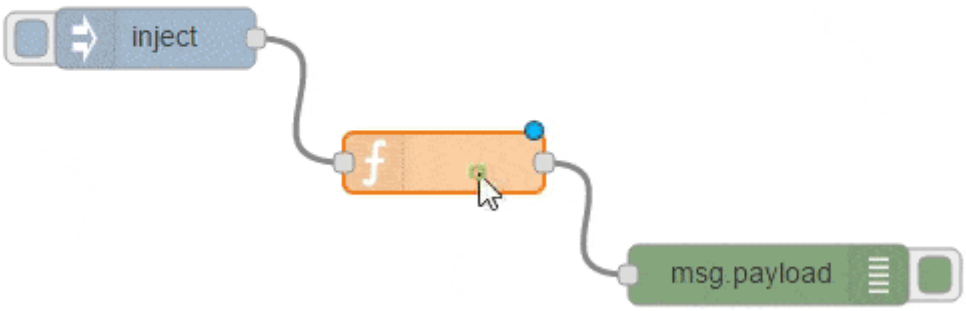
نموذج البرمجة Node-RED

كما وضح في الدروس السابقة، يستخدم Node-RED التدفق المرئي القائم على نموذج البرمجة. هذه ليست صفة فريدة لـ Node-RED . فقد استخدم نهج مماثل في العديد من المجالات الأخرى مثل الموسيقى و الوسائط المتعددة (Max MSP) ، ولعب الأطفال (Lego Mindstorms) ، والأتمتة الصناعية (LabVIEW) . مثل هذه الأدوات، Node-RED يجعل من السهل التنقل بين التصميم و التطوير لمهام الاندماج السريع ، النماذج و التنمية . وقد تم الإطلاع على كيفية إنشاء تدفقات و ربط العقد معا، وبعض أساسيات البرمجة في Node-RED .

في هذا الدرس، سوف نلقي نظرة أكثر تفصيلا على نموذج البرمجة لـ Node-RED و بعض المفاهيم الأساسية، وإكتشاف تفاصيل نموذج تدفق الرسالة الذي يكمن وراء Node-RED . ثم التعمق في كيفية برمجة عقد function باستخدام الجافاسكريبت (JavaScript)و إنشاء التدفقات الفرعية التي يمكن إعادة إستخدامها لإضافة المهام الخاصة بك إلى مجموعة من العقد المتوفرة مع Node-RED .

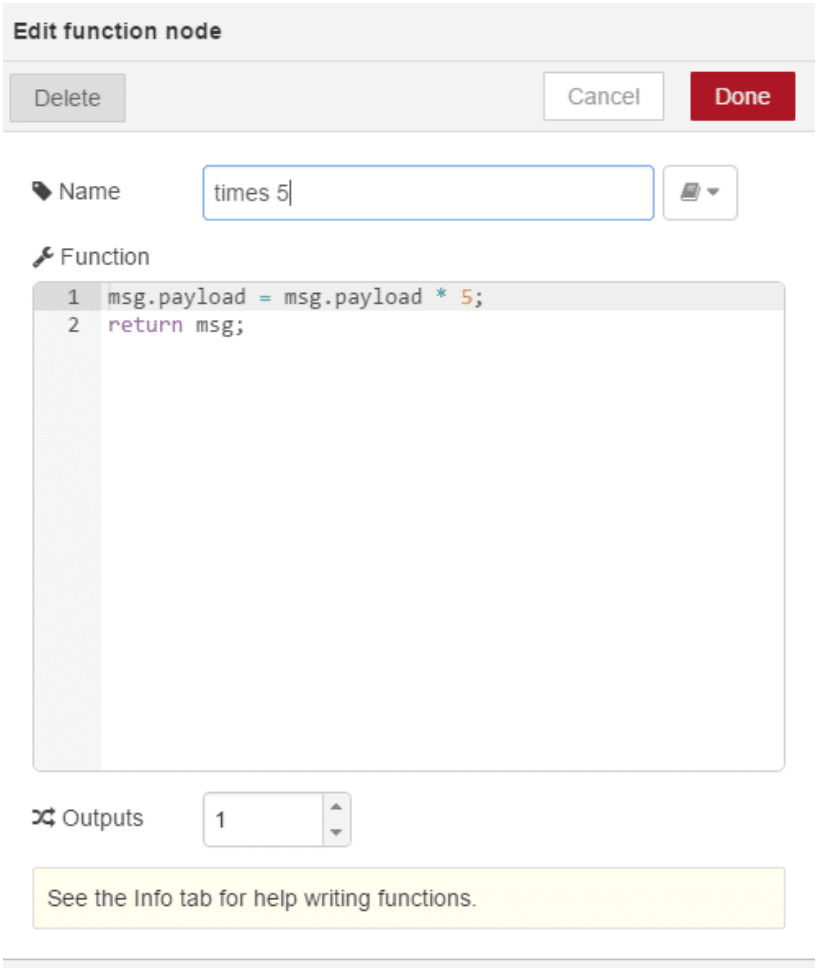


نحن نستخدم نسخة استضافة سحابة من Node-RED عن طريق FRED . قم بالتسجيل للحصول على حساب مجاني في FRED .

للإطلاع على المزيد حول FRED من هنا.

عقد الـ Function :

عقد الـ Function من العقد التي يمكنك استخدامها عندما لا يكون هناك عقدة موجودة مخصصة لمهمتك في متناول اليد. باستخدام عقدة Function، يمكن ان تكتب بنفسك كود جافا سكريبت (javaScript) الذي يعمل على الرسائل التي مرت و إرجاع الرسائل إلى العقدة التالية للمعالجة. لكتابة عقدة function ، يمكنك كتابة دالة جافا سكريبت (javaScript) باستخدام محرر الكود المدمج كما هو مبين بالصورة أدناه :



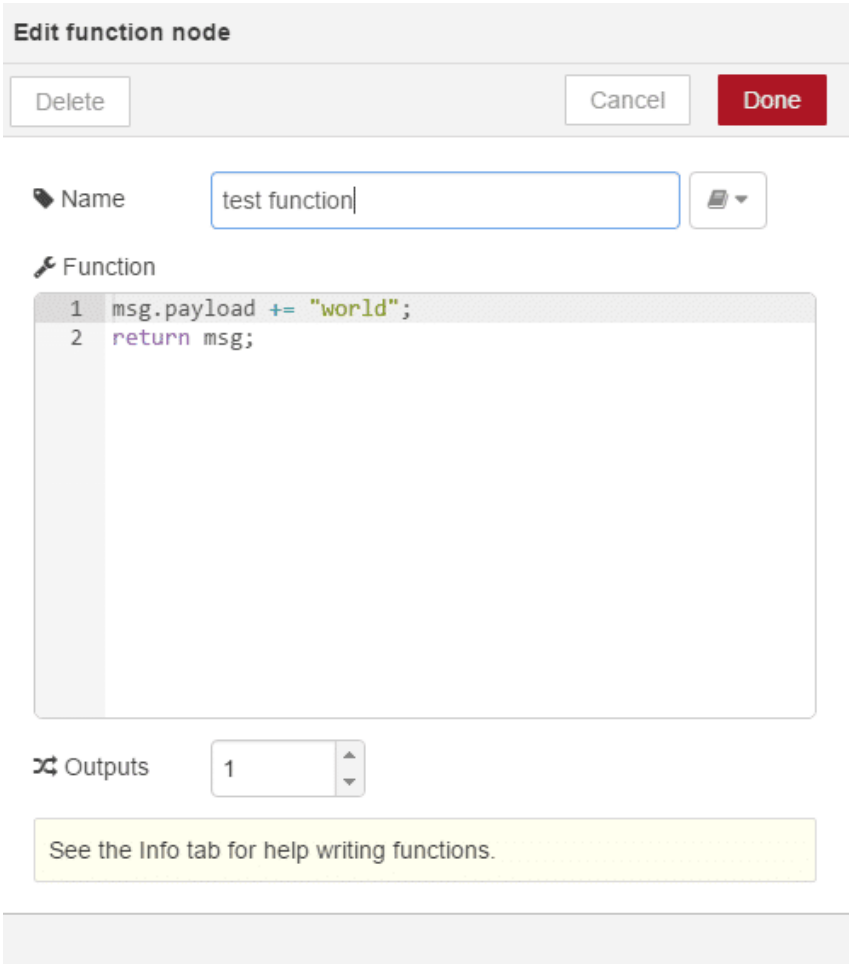
الكتابة على عقدة Function :

دعونا نبدأ كتابة عقد Function . سوف نقوم بإنشاء تدفق بسيط من عقدة inject و عقدة debug كما هو موضح بالصورة

أبسط عقدة Function هي التي ترجع Null . عندما يتم ترجيع null ، لا يتم تمرير رسالة إلى العقد التالية في التدفق و التدفق ينتهي هنا. لإنشاء العقدة التي تمرر الرسالة "كما هي" ، يمكنك إرجاع الرسالة نفسها . وهذا الكود يكون في العقدة افتراضيا، كما هو موضح أدناه :

```
return msg;
```

دعونا نضيف بعض المحتوى إلى الحمولة، في هذا المثال، و المجموعة التالية من الأمثلة التي ستستكشف كتابة كود جافا سكريبت (JavaScript) لعقدة function ، سوف تستخدم نفس التدفق في الصورة أعلاه وعن طريق تعديل عقدة function لتغيير كود الجافا سكريبت باستخدام محرر عقدة function، كما هو موضح بالصورة ادناه :



كود عقدة Function يعمل على إضافة نص للحمولة :

```
msg.payload += "world";
return msg;
```

في هذا الكود، سيتم إضافة سلسلة "world" إلى حمولة الرسالة الواردة (السطر الأول). ثم يتم إرسال هذه الحمولة المحدثة. وذلك باستخدام الإرجاع (return) ، كرسالة مخرجة (السطر 2)إلى العقدة التالية في التدفق

عند كتابة الكود أعلاه في محرر عقدة function ، عند نشرة و اختبار التدفق سترى في لوحة الإخراج عند تبويب debug الطابع الزمني لعقدة inject و النص "world" ملحق به.

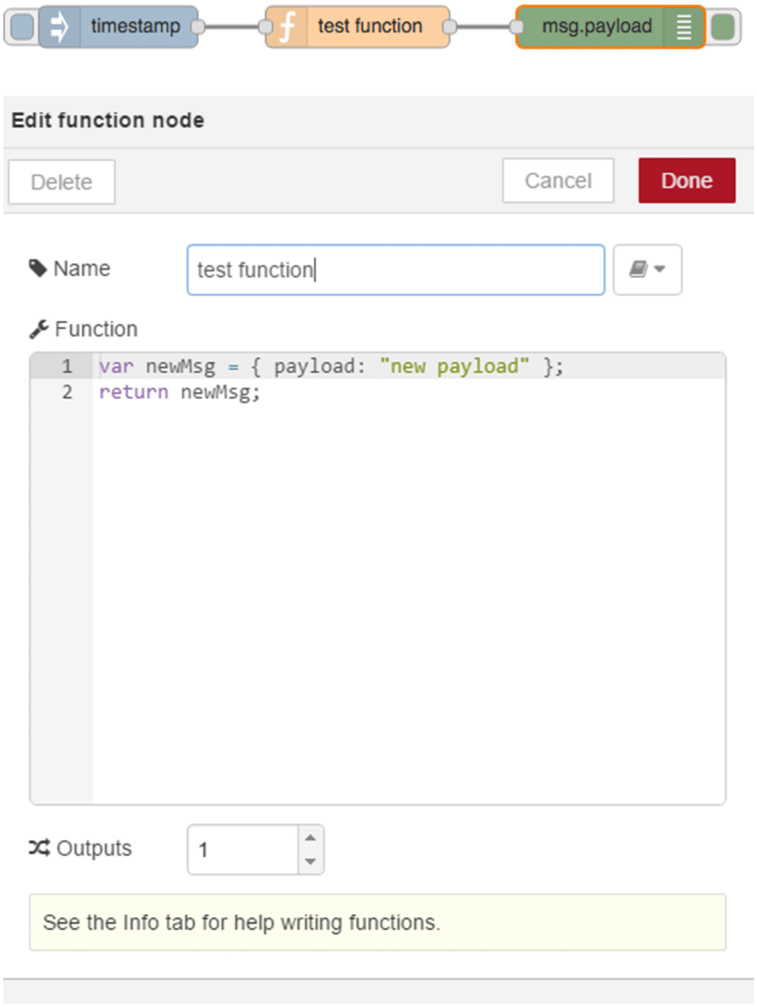
إذا قمت بتعديل عقدة inject لضبط سلسلة (string) بدلا من الطابع الزمني (timestamp) ، وتعيين السلسله لـ "hello" ، سوف بتظهر لك الرسالة "hello worls " في لوحة الإخراج عند تبويب debug عند نشر واختبار التدفق .

إنشاء و إرجاع رسالة جديدة في عقدة Function :

في الكود أدناه يبين كيفية إنشاء رسالة جديدة عن طريق تعريف رسالة جديدة newMsg ، وتعيين سلسلة "new payload" إلى خاصية الحمولة (payload) (في السطر 1), ومن ثم إرجاع الرسالة الجديدة في السطر 2.

```
var newMsg = { payload: "new payload" };
return newMsg;
```

قم بالتعديل على محرر عقدة function في التدفق لكتابة الكود كما هو موضح بالصورة التالية :

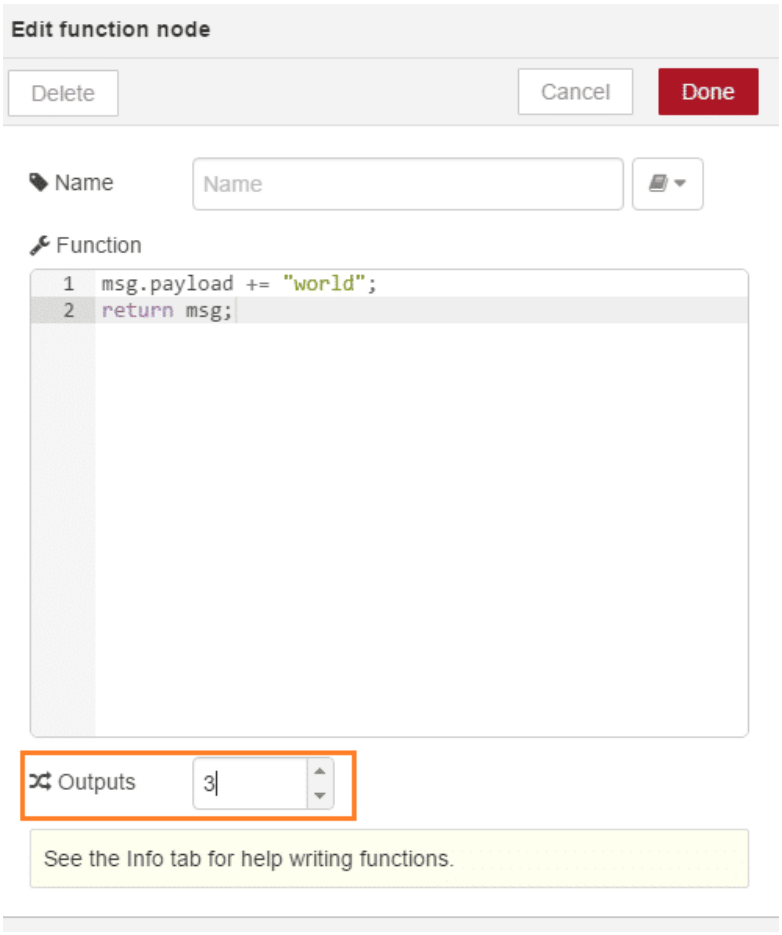


فعند نشر التدفق و الضغط على زر عقدة inject ، سيتم إخراج الرسالة الجديدة التي تم إنشائها لتصل إلى عقدة debug .

إنشاء و ارجاع رسائل متعددة في عقدة Function :

عقدة Function تسمح ايضا لإعدادها مع نواتج متعددة. وبمجرد إعداد نواتج متعددة، يمكنك إرسال رسائل متعددة، واحدة لكل مخرج، وذلك باستخدام مجموعة (array).

أولا، دعونا تعديل عقدة function ليصبح للعقدة ثلاثة نواتج، وذلك باستخدام إعداد العقدة كما في الصورة ادناه :

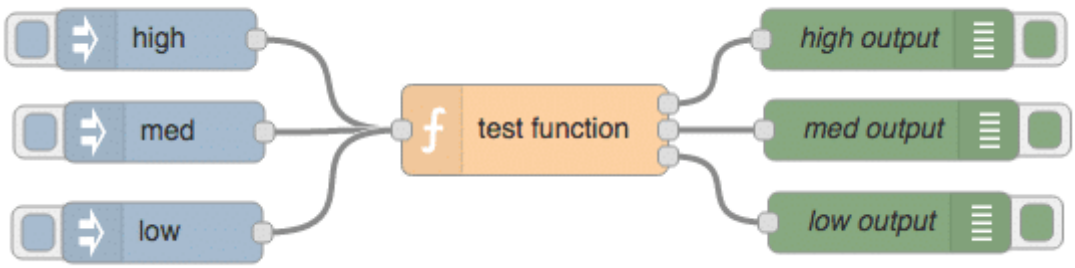


الكود ادناه يوضح كيفية إرسال رسالة إلى واحد من ثلاثة نواتج مختلفة على أساس قيمة الحمولة "low","med","high"

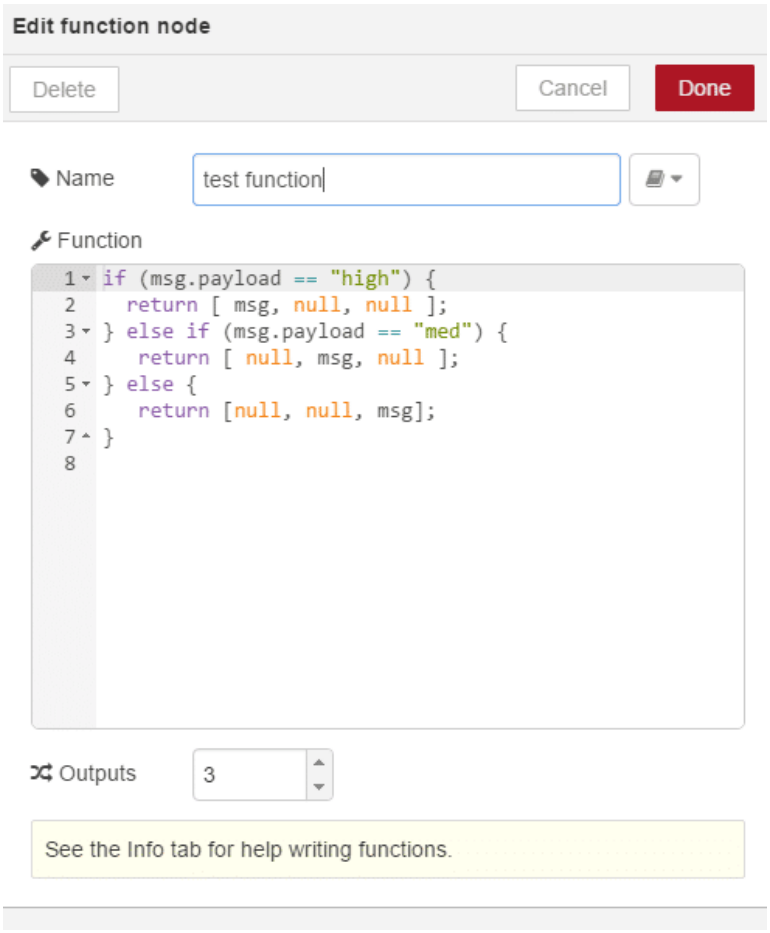
```
if (msg.payload == "high") {
return [ msg, null, null ];
} else if (msg.payload == "med") {
return [ null, msg, null ];
} else {
return [null, null, msg];
}
```

السطر الأول يقوم بالتحقق من الحمولة القادمة إذا تم تعيينها على أنها "high" ، اذا تحقق الشرط يتم تمرير الرساله على المخرج الأول و رسالة null على المخارج الأخرى ، من خلال return [msg, null, null]. و " med" يمرر الرسالة على المخرج الثاني و اي شي آخر يتم إرجاعة على المخرج 3 .

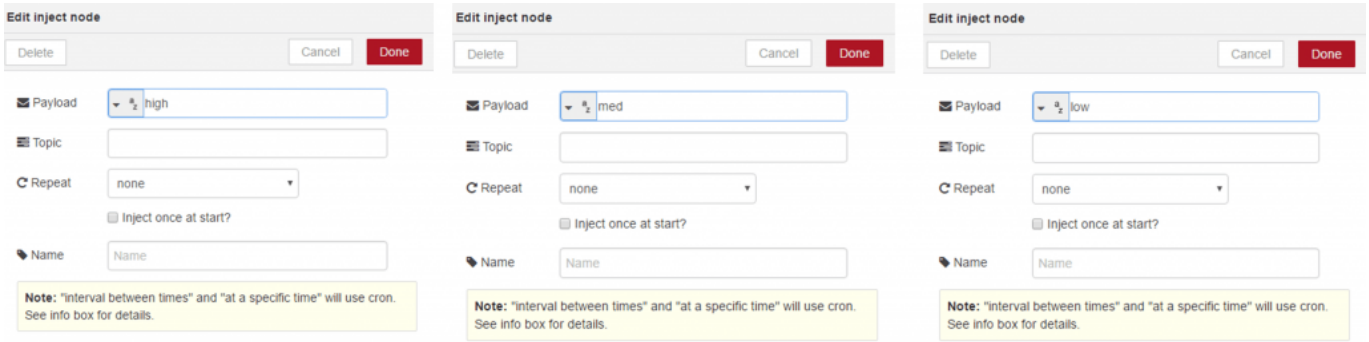
لتجربة ذلك ، قم بإيصال ثلاث عقد inject و ثلاث عقد إخراج كما هو مبين بالشكل أدناه .



قم بالتعديل على كود عقدة الـ function كما هو موضح أدناه :



قم بإعداد عقدة inject لتعيين سلسلة نصية إما high ، med ، low ، قم بنشر و اختبار التدفق .



حيث انه من الممكن إرسال رسائل متعددة من عقدة function في التسلسل. للقيام بذلك ، للقيام بذلك، قم بالاطلاع على الكود ادناه ، يوضح كيفية كتابة دالة تقوم بإرجاع 10 رسائل جديدة تحتوي على حمولة عدد من 0 إلى 9 . في السطر الأول يتم إنشاء المجموعة array الجديدة باسم msgList لعقد الرسائل. في السطر 2 و 3 عبارة عن loop حلقة 10 مرات لإضافة عنصر جديد إلى مجموعة msgList . و أخيرا ، السطر 5 يقوم بإرجاع المصفوفة array ، والتي تحتوي على عنصر و الذي هو بحد ذاته مجموعة من 10 عناصر . كما رأيتم في المثال السابق، من خلال ارجاع المصفوفة array من عنصر واحد ، فبالتالي يتم ارسال الرسالة غلى المخرج 1 من العقدة. وبما ان العنصر يحتوي على 10 عناصر ،فإن Node-red يرسل كل عنصر على شكل سلسلة من الرسائل المتتابعة على المخرج 1 .

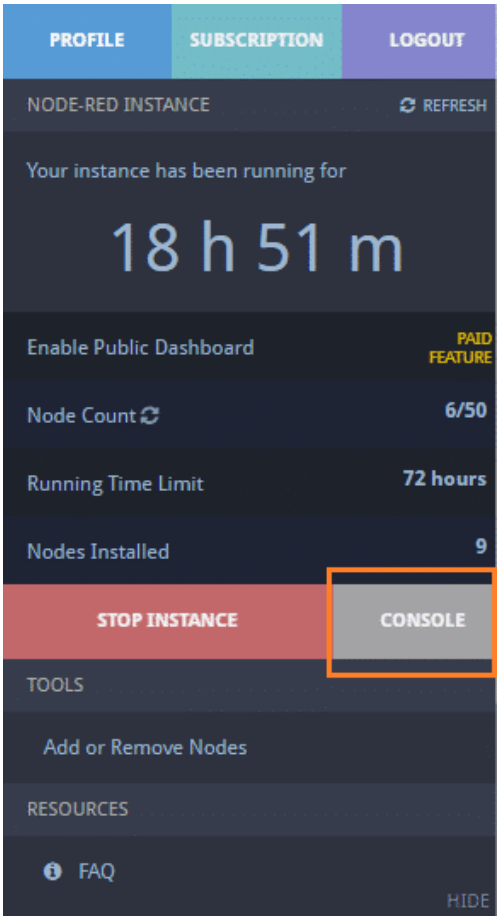
```
var msgList = [];
for (var i=0; i<10; i++) {
msgList.push({payload:i});
}
return [msgList];
```

عقد Functions :

كود عقدة function الخاصه بك لديها امكانية للوصول إلى عدد قليل من الوظائف المضمنة في وحدة العقدة . وهذه تسمح لك لتسجيل النص إلى وحدة الإخراج أو التصحيح debug و ارسال الرسائل عن طريق استدعاء الـ function بدلا من إرجاعهم من عقدة function نفسها .

node.log() – تسجيل الرسائل إلى وحدة التحكم
node.warn() – تسجيل الرسائل إلى وحدة التحكم ورسالة التحزير في لوحة الإخراج عند تبويب debug .
node.error() – تسجيل إلى وحدة التحكم و الخطأ في لوحة الإخراج التصحيح debug .

في FRED ،يمكن عرض وحدة التحكم (console) لـ Node-RED من خلال الذهاب إلى الصفحة المقصودة، ثم النقر على قائمة Console. حيث أنه يمكن أن يكون التنقل بطيء بين محرر NR ووحدة التحكم، بعد عرض وحدة التحكم ، قم بالضغط على زر تحديث لمعرفة التحديثات إلى وحدة التحكم (console) . كما ان الوظائف node.warn و node.error ايضا هم مخرجات على لوحة الإخراج في تبويب debug ، فإنه من المفيد في بعض الأحيان الاستفادة من هذه الوظائف لتصحيح الأخطاء عندما تكون وحدة التحكم غير متوفرة.



وبالإضافة إلى وظائف التسجيل logging و تصيحي الأخطاء debugging ، وظيفة node.send() يمكن استخدامها لإرسال رسائل إلى العقدة في التدفق باستخدام وظيفية الاستدعاء، بدلا من إرجاع الرسائل . الكود أدناه يوضح كيفية استخدام node.send() في عقدة function ليرسل رسالة بعد فترة محددة من الوقت .

```
// send message after 10 second delay
setTimeout(function() {
```

```
node.send(msg);
}, 10000);
return null;
```

وتشمل وحدات أخرى متوفرة لعقد function ، حيث تشمل ما يلي :

console – في حين تتوفر وحدة التحكم (Node.js (consol ، إلا أن node.log() هو الأسلوب المفضل للتسجيل.
util – وحدة Node.js util . يقدم هذه الوحدة خدمات مفيدة مثل وظائف سلسلة الشكل وتوليد تمثيل سلسلة من الكائنات.
Buffer – وحدة Node.js Buffer هي للتعامل مع البيانات الثنائية من TCP streams او من الملفات .

السياق (Context) :

كما تعلمت أن الرسائل هي الطريقة الوحيدة للحصول على بيانات الدخول و الخروج من العقد . في حين أن هذا غير صحيح عموما، هناك استثناء واحد لهذه القاعدة وهي متاحة لعقدة Function . عقد funcrion لها قدرة على الوصول إلى كائن خاص يسمى السياق (context) و الذي يتم استخدامه لحفظ البيانات في الذاكرة التي تستمر من رسالة إلى وصول الأخرى. وهذا أمر مهم للعقد التي تحتاج إلى الحفاظ على مؤشر أو العقد أو مجموع البيانات في الرسائل. وبالإضافة إلى هذا السياق المحلي، يوجد السياق العالمي context.global وهو متاح لتبادل البيانات بين كل من عقد function من التدفق. وسيتم تغطية استخدام السياق عند مناقشة عقدة function في مزيد من التفاصيل .

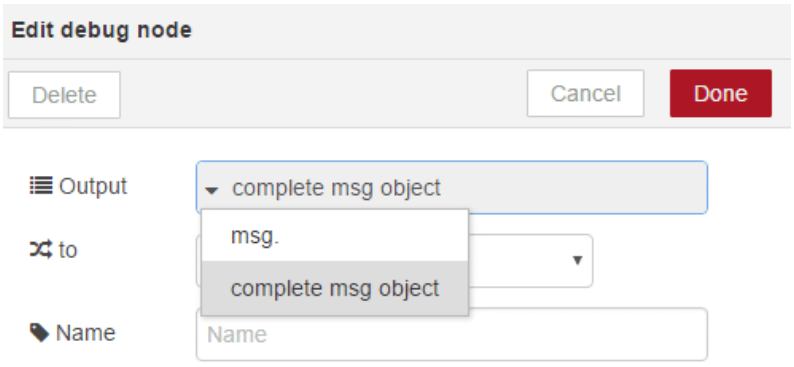
استخدام السياق (Context) في عقدة function :

وحدة خاصة تسمى السياق (Context) ، تستخدم لتخزين البيانات بين استدعاء الـ function ، ويتوفر على عقدة function . وهي مفيدة عندما تحتاج وظيفية للإحتفاظ بالـ state للقيام بالمعالجة . على سبيل المثال، دعونا نستخدم السياق (context) لحساب عدد الرسائل التي تمت معالجتها في عقدة function عندما تم نشرها. دعونا ننشئ عقدة function التي تضيف خاصية العد إلى كل رسالة تقوم بمعالجتها، كما هو موضح بالكود ادناه .

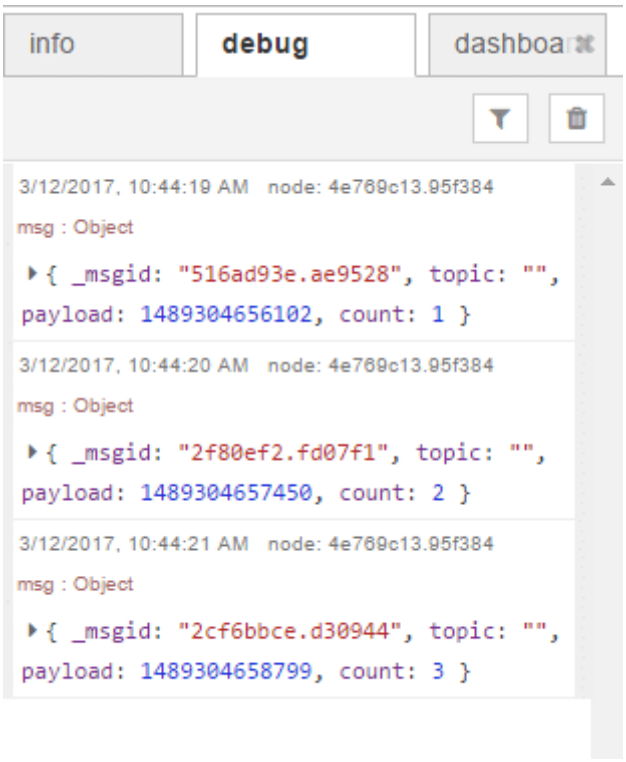
```
if (!context.value) {
context.value = 0;
}
context.value +=1;
msg.count = context.value;
return msg;
```

ويمكن التوصيلها، عادة يتم بناء التدفق كما هو موضح بالصورة أدناه :

تأكد من تغيير عقدة debug لعرض 'complete msg object' حتى تتمكن من رؤية خاصية العد.



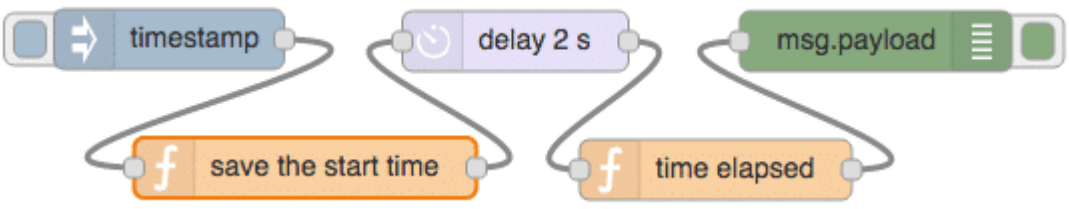
قم بنشر (Deploy) و اختبار التدفق



ستلاحظ ان السياق يتم إعادة تعيينه كل مرة يتم نشر عقدة Function . إذا كنت بحاجة لحفظ الـ state بين عمليات النشر ، وسوف تحتاج إلى استخدام بعض وحدات التخزين الخارجية مثل ملف أو قاعدة تخزين .

استخدام السياق العالمي (Global Context) :

بالإضافة إلى سياق عقدة Function الفردية ، وحدة السياق العالمية هي متاحة لمشاركة السياق بين عقد Function . دعونا نستخدم هذه الوحدة لمعرفة مدى دقة عقدة التاخير (delay) ، كما هو مبين بالشكل التالي :



يظهر الكود لعقدة Function لحفظ وقت بدء التدفق في خاصية السياق العالمي أدناه :

```
context.global.startTime = new Date().getTime();
return msg;
```

وفي عقدة الـFunction الأخرى ، يتم احساب الوقت المنقضي، حيث نقوم بإنشاء رسالة تحتوي على الوقت الذي انقضى منذ مرور الرسالة من خلال عقدة Function لحفظ وقت البدء ، كما هو مبين بالكود أدناه :

```
var currentTime = new Date().getTime();
var timeElapsed = (currentTime - context.global.startTime)/1000;
msg.payload = "Time elapsed is: "+timeElapsed+"s";
return msg;
```

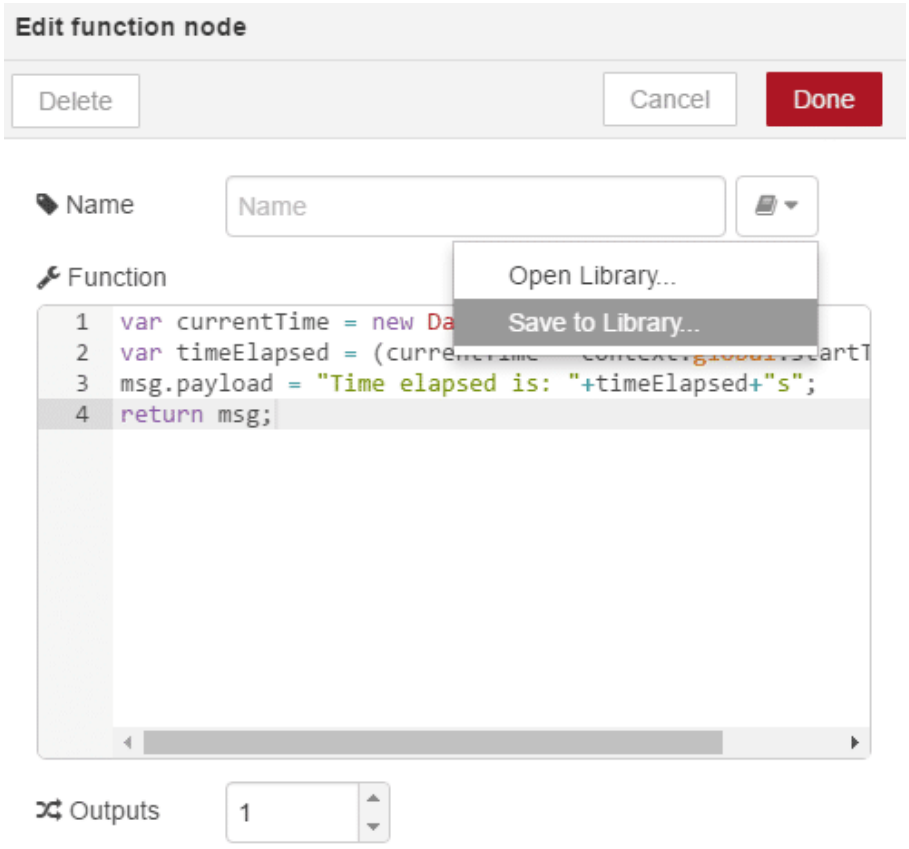
إضافة و إعداد عقدة التأخير (delay) لتأخير الرسالة لمدة 2 ثانية ، ثم انقر على عقدة inject ، والانتظار حوالي 2 ثانية . إذا سارت الأمور بشكل جيد، فإن إخراج عقد debug يجب أن تشير إلى الوقت المنقضي وهو على مقربه من 2 ثانية .

وميزة هذا النهج هي أننا لا نحتاج رسالة لتحمل وقت البدء. و العيب هو أنه وصلت رسالة أخرى في حفظ وقت البدء قبل الوقت المنقضي ، سيتم الاستبدال و الكتابه على المتغير context.global.startTime .

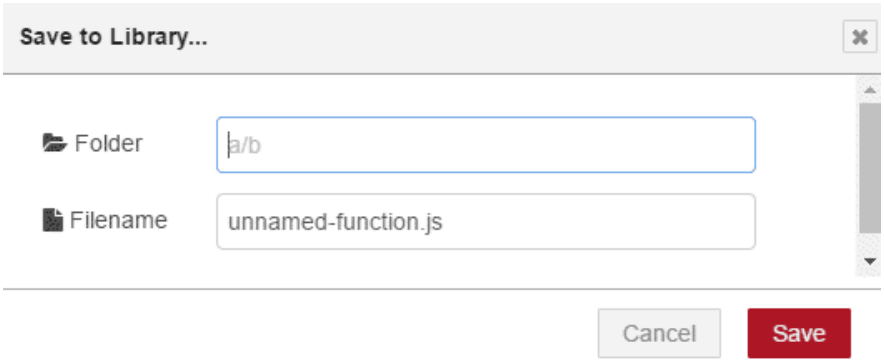
حفظ كود Function الخاص بك في مكتبة :

بمجرد أنك انشأت بعض التعليمات البرمجية للاستخدام في عقدة function ، غالبا ما يكون مفيد حفظ هذا الكود لإعادة استخدامه في وقت لاحق. وثمة طريقة للقيام بذلك وهو قص و لصق الكود إلى function جديدة، ولكن لتوفير الوقت، يمكنك استخدام مكتبة عقدة function التي تم إنشاؤها في Node-RED .

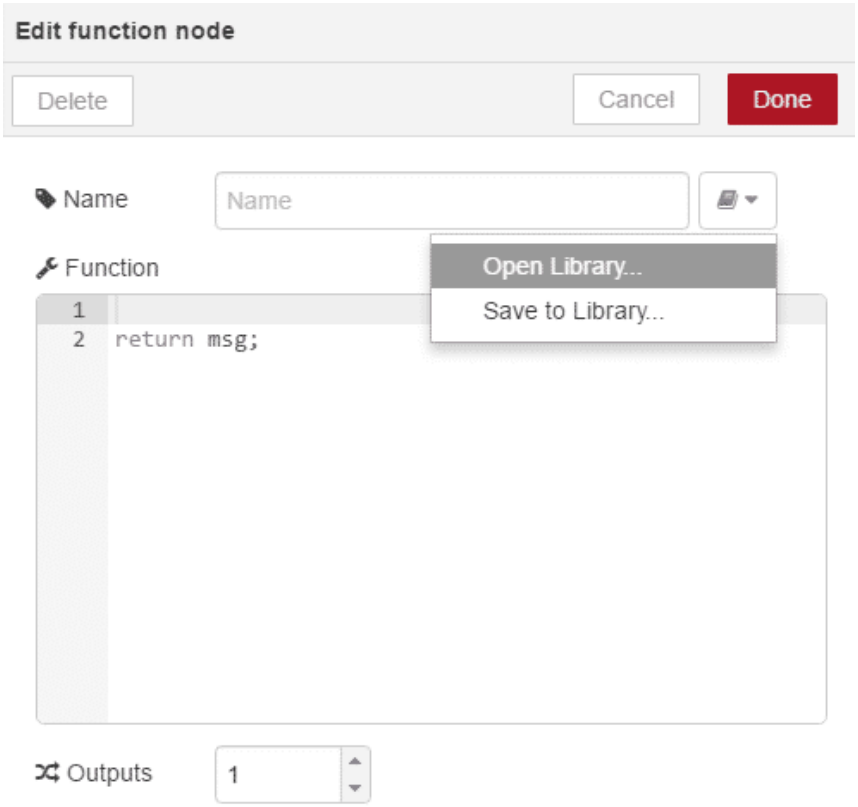
لحفظ الكود الخاص بك في مكتبة الـ Function ، انقر على ايقونة الكتاب في حوار إعدادات عقدة function على الجانب الأيمين لمربع تحرير اسم النص (Name) ، ثم اختر "Save to Library" حفظ إلى المكتبة .



يمكنك بعد ذلك توفير اسم المجلد و اسم الملف لحفظ الكود كما هو مبين بالشكل أدناه :



للاستفادة من التعليمات البرمجية الخاصة بك يمكنك إنشاء عقدة function جديدة، ثم النقر على أيقونه الكتاب ، ثم الضغط على "Open Library..." مكتبة مفتوحة ثم إختيار مكتبة الكود المحفوظة.



سيتم تحميل التعليمات البرمجية في مربع الحوار. إذا كان هذا هو الكود الذي تريده. انقر فوق موافق (OK) ، وسوف تضاف إلى عقدة Function الخاص بك. يمكن بعد ذلك تعديلها حسب الحاجة .