Node-RED

Cheat Sheet

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire things together using the wide range of nodes in the palette that can be deployed in a single-click.

Getting Started – Running Locally

online documentation: http://nodered.org/docs/getting-started/installation

Step 1 – Install NodeJS (https://nodejs.org/)

Step 2 – Open the command line (Windows) or Terminal (Mac):

Windows users will need to run the command line as an admin user (right click - "Run as Adminstrator")

Step 3 – Run the command:

Windows: `npm install -g --unsafe-perm node-red`

Mac: `sudo npm install -g --unsafe-perm node-red`

Step 4 – From the command line, you can now run the command `node-red` and Node-RED should start. Navigate to http://localhost:1880/ to use it.

Getting Started – Running on Bluemix

online documentation: http://nodered.org/docs/platforms/bluemix

Step 1 – Login or sign-up for an account at Bluemix.net (https://console.ng.bluemix.net/)

Step 2 – Navigate to the "catalog" *(option in top-right of screen)*

Step 3 – Search "Node-RED" and select the "Node-RED Starter" boilerplate

Step 4 – Entire the name of your Node-RED application, in addition to a host name (which must be all lower case and contain no spaces).

Step 5 – Click "Create"

Step 6 – Wait a few minutes for the application to deploy. Once complete, navigate to https://your-host-name.mybluemix.net.

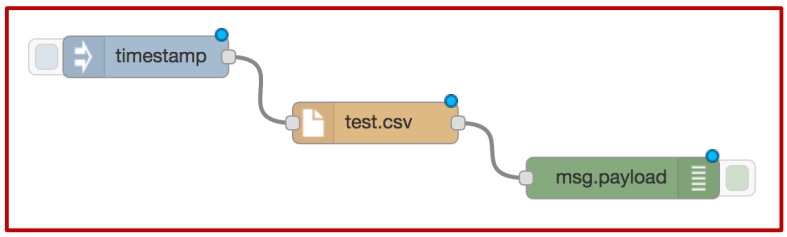
Note: Double click any node to edit its properties.

Getting Data - Local File

only available when running Node-RED on your local machine

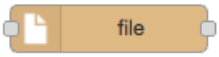
A lot of the challenges will require you to process and analyse the contents of downloaded files, such as a *.csv* file.

In Node-RED, this can only be done when running locally, but it's easy to do:

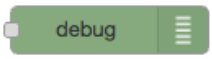


Components

The **inject** node is what will trigger the reading of the file. Click the button  to run this flow.



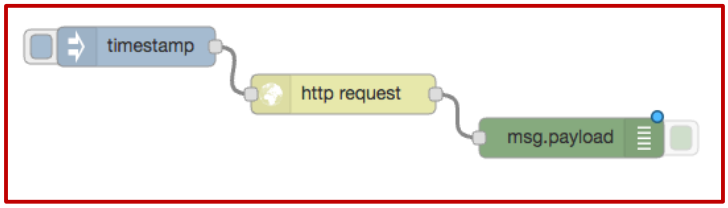
The **file** node provides the option to define a file name. The file must be placed in the directory (or a sub-directory) that you ran the `node-red` command from.



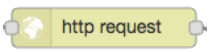
A common way of testing your flow in Node-RED is using this **debug** node. Anything nodes connected to here will cause the **debug** node to output the message's content to the "Debug" panel on the right-hand side of the Node-RED interface.

Getting Data – API Requests

A common method for getting access to remotely stored data is via an API. To use an API, you're often provided with a URL, and when you navigate to that URL, the data is returned.



Components



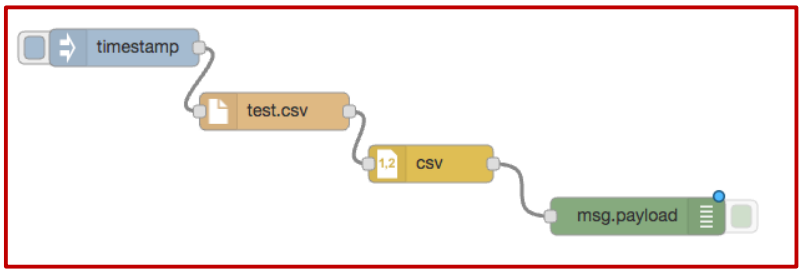
The **http request** node provides the option to define a URL. When run, the node will request the resources at the URL and outputs the response content.

The other nodes (**inject** and **debug**) are defined above in the *"Getting Data – Local File"* section.

Converting local .csv files

only available when running Node-RED on your local machine

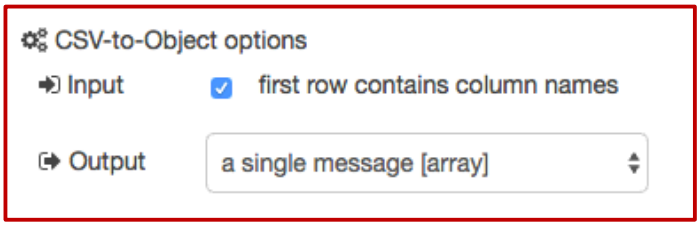
An easy way to process data from a *.csv* file is to first convert it into *.json* format, this then make it easy to use with such nodes as the **function** node.



In this example we have a *.csv* file which contains the following:

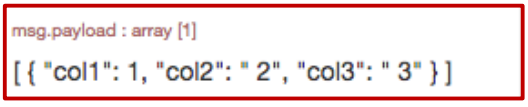
```
col1, col2, col3
1, 2, 3
```

We're going to use the exact same components from the previous "reading a file" example, but add in a csv node which will convert the data into a digestible format.



In this case, the *.csv*, contains columns headers (a row detailing what each column title is) and so we need to ensure the "*first row contains column names*" option is selected, we're also going to convert this data into a single message, and so we select the "*output -> a single message (array)*" option.

The output of this (show in the Debug Panel) would be the following:

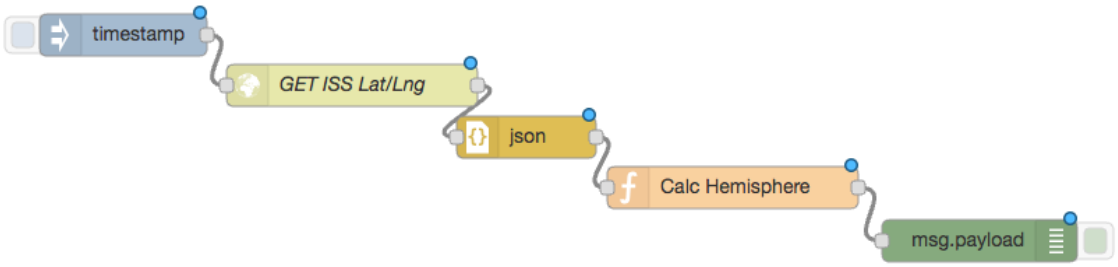


This data is now in a *.json* format, which can be used as in the *"Analysing Data"* section.

Analysing Data

An easy way of analysing and processing data in Node-RED is by using the **function** node. The **function** node allows you to write **JavaScript** to process the incoming message, and define the content that is passed onto the rest of the flow.

Let's take an example of analysing the data from a **http request**. In this example, we are going to do a **http request** to http://api.open-notify.org/iss-now.json, which will give us the live latitude and longitude that the International Space Station is over.



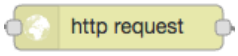
The function inside the "Convert Data" function node would be as follows:

```
var pos = msg.payload.iss_position;

if (parseInt(pos.latitude) > 0) {
    msg.payload = "Northern Hemisphere";
} else {
    msg.payload = "Southern Hemisphere";
}

return msg;
```

In this case, the flow will result in the text "Northern Hemisphere" or "Southern Hemisphere" being printed into the Debug Panel, depending on the current ISS location.

Components

(see *"Getting Data – Local File"* example)

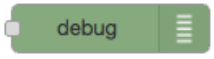


(see *"Getting Data – API Requests"* example)

The **json** node converts the text/string output from the **http request** and turns it into a *.json* object that can be easily read by the **function** node.

The function node allows you to write your own JavaScript to manipulate and analyse the incoming message, `msg.payload`.

By defining `msg.payload` and returning `msg` from the function, that will set a new value for the message and pass that on to the nodes connected to the output of this **function** node.



(see *"Getting Data – Local File"* example)

Using Watson

online documentation: http://ibm.biz/noderedlabs