

Fundamentals of IoT Software © 2022 by Luca Mottola
is licensed under CC BY-NC 4.0



To view a copy of this license, visit
creativecommons.org/licenses/by-nc/4.0/

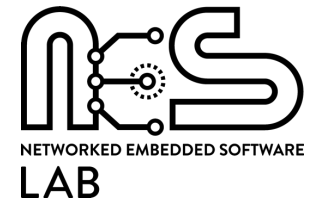




POLITECNICO
MILANO 1863



POLITECNICO
MILANO 1863



Node-RED Fundamentals

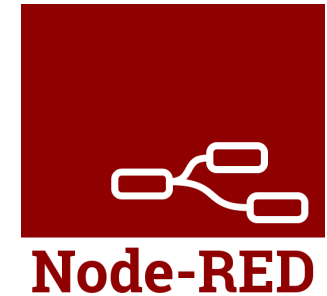
Luca Mottola

`luca.mottola@polimi.it`

(version 0.1)

Outline

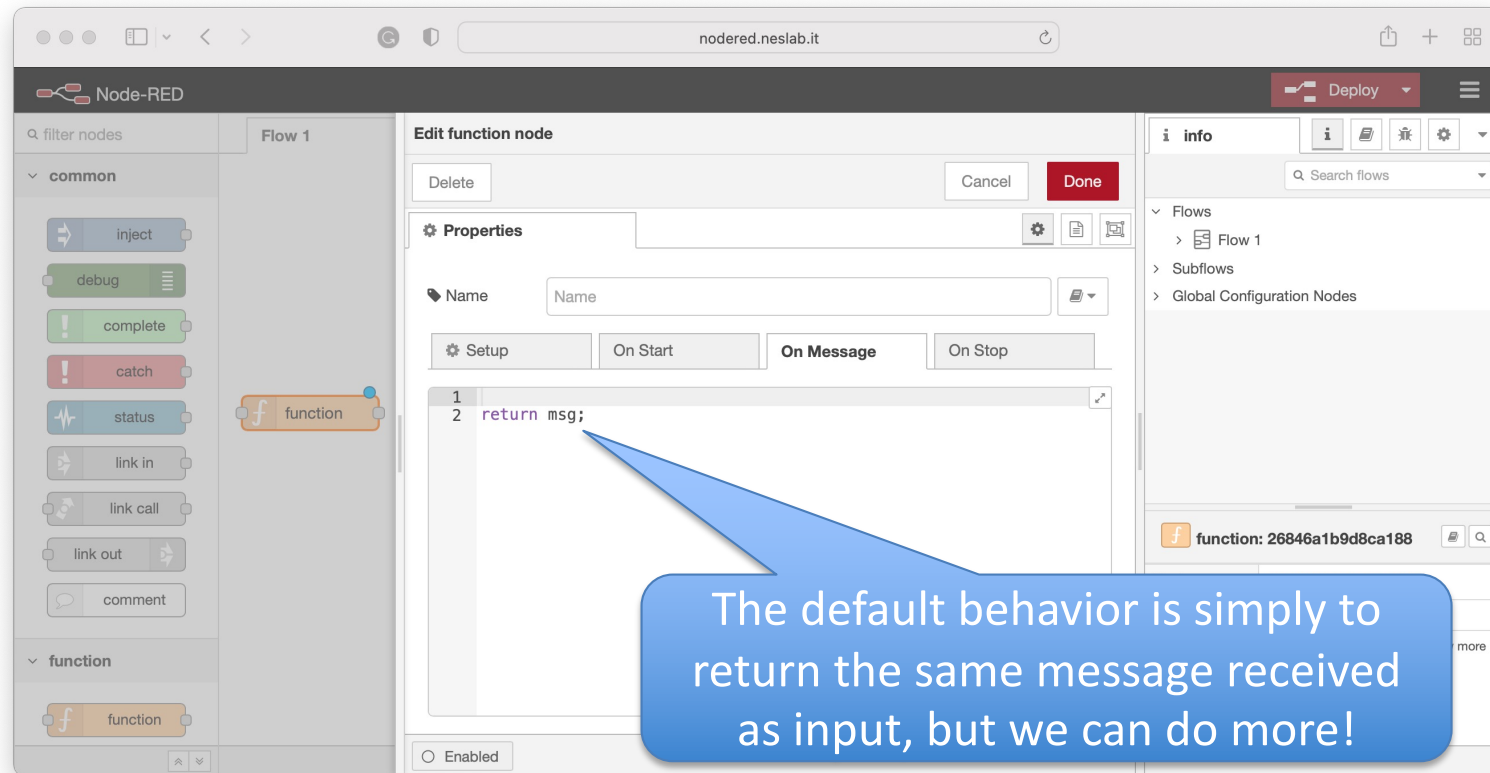
- Basics
- Function nodes
- Data sharing



Function Nodes



Function Nodes (1/3)



- Function nodes are generic **containers for JavaScript code** that
 - Receive a message object as **input**
 - Generate one or more message objects as **output**



Function Nodes (2/3)



```
if (msg.payload % 2==0) {  
    msg.payload = "Time is even";  
} else {  
    msg.payload = "Time is odd";  
}  
return msg;
```

Plain JavaScript code!



Function Nodes (3/3)

The screenshot displays the Node-RED web interface. On the left, the 'common' and 'function' node palettes are visible. A 'function' node is selected and placed in the workspace. The 'Edit function node' dialog is open, showing the 'On Message' tab. Three blue callout boxes provide context:

- Code to be executed every time a message is received**: Points to the 'On Message' tab.
- Code to be executed when the flow is deployed**: Points to the 'Setup' tab.
- Code to be executed when the flow stops**: Points to the 'On Stop' tab.

The 'On Message' tab contains the following code:

```
1  
2 return msg;
```

The 'Setup' tab is currently selected. The 'On Stop' tab is also visible. The 'Enabled' checkbox is checked at the bottom of the dialog.

On the right side of the interface, the 'Subflows' and 'Global Configuration Nodes' sections are visible. Below them, a node information panel shows:

function: 26846a1b9d8ca188	
Node	"26846a1b9d8ca188"
Type	function

show more ▼



Function Node Patterns (1/2)

- Override the payload of the incoming message

```
let newMsg = {  
  topic: msg.topic,  
  payload: "New message!",  
  _msgid: msg._msgid,};  
return newMsg;
```

The new message is returned instead of the original `msg`

The payload is overwritten, no matter what was the payload of the original `msg`



Function Node Patterns (2/2)

- Slicing the payload of the incoming message

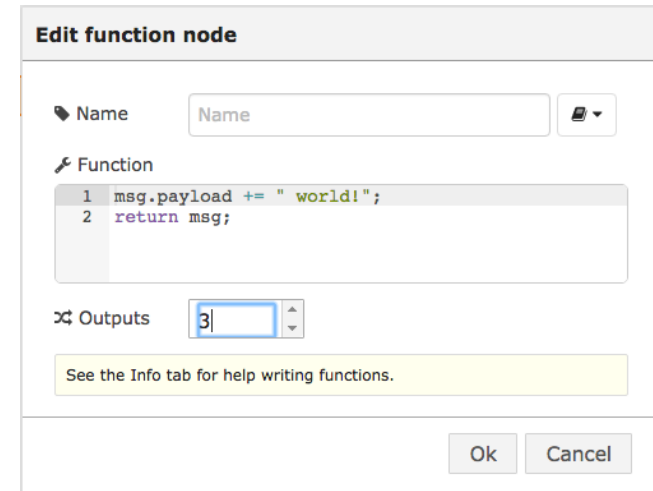
```
let newMsg = {  
  topic: msg.topic,  
  payload: msg.payload.temperature;  
  _msgid: msg._msgid,};  
return newMsg;
```

Assuming the payload is itself an object with a **temperature** key, the new message takes that value and makes it the new payload!



Multi-output Function Nodes

- Function nodes may determine **where** messages flow!
 - Using **JavaScript** arrays
 - The size of the array must be the same as the number of outputs configured for the function node



Edit function node

Name

Function

```
1 msg.payload += " world!";  
2 return msg;
```

Outputs

See the Info tab for help writing functions.

Ok Cancel

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

The **null** value in the array generates no messages on the corresponding output



Multi-message Function Nodes

Creates an array of 10 messages and returns it as a sequence in place of the original `msg`

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

- Function nodes may output multiple messages as arrays
- **Note:** function nodes may be saved and loaded to/from the node library!

