Node-RED Fundamentals

Luca Mottola
`luca.mottola@polimi.it`

(version 0.1)

Outline

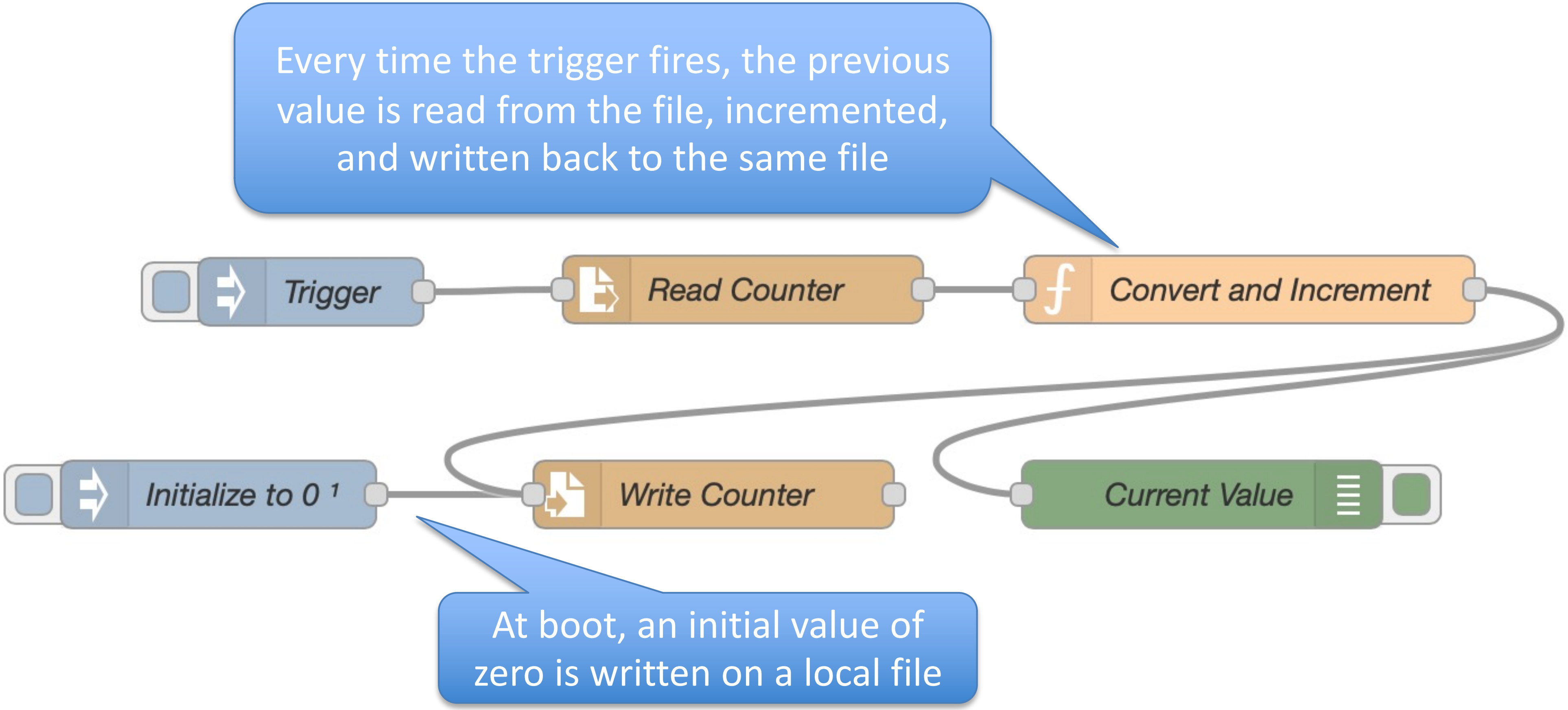
- Basics
- Function nodes
- Data sharing

Data Sharing

Data Sharing

- Often, Node-red applications cannot be encoded in a single flow execution
- Data must be shared
  - Across different executions of the same block
  - Across different executions of the same flow
  - Across different flows
- Files are **one way** to achieve this

Example: Counter



Every time the trigger fires, the previous value is read from the file, incremented, and written back to the same file

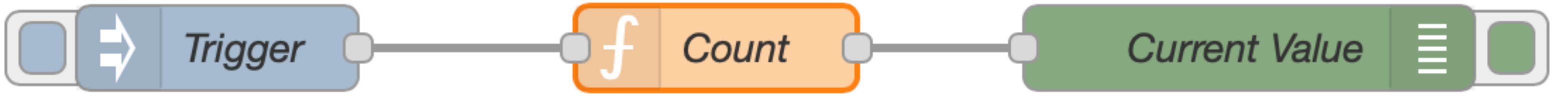
At boot, an initial value of zero is written on a local file

Find the code at: bit.ly/3J1i3QI

Node Context

- A special `context` object exists **for function nodes** to retain state across invocations
- It only applies to individual nodes
- The context is **re-initialized** when deploying the flow
  - You generally need to use some external storage (a file, a db, …) if you want state to be retained across deployment actions
  - You can also change the back-end of the context store in Node-RED configuration file

Example: Counter Revisited



The function node uses node context to store the value of the counter!

A property named **`value`** is defined at the first execution and then used as the counter

```
if (!context.get(”value”)) {
  context.set (”value”,0);
}
msg.payload =
  context.get(”value”);

context.set(”value”,
  context.get(”value”)+1);
return msg;
```

Find the code at: bit.ly/3HQvbql

Flow and Global Context

- A `context` module also exists for sharing data across the entire **flow** or **globally** across flows of the same Node-RED instance
  - You may access these in JavaScript code using `flow` or `global`, instead of `context`
- The same limitations related to initialization and deployment apply as node `context` modules