

# Develop A Web Application Using Node-RED Service.

The image displays two screenshots of the Node-RED web interface, illustrating the development of a web application using the Node-RED Service.

**Top Screenshot:** Shows the main Node-RED workspace with a flow named "Flow 1". The flow includes an "IBM IoT" node connected to "Temperature" and "Humidity" nodes, which are then connected to "msg.payload" nodes. A "debug" sidebar on the right shows the output of the flow, including a JSON object: 

```
{ "temperature": 34, "humidity": 74, "gaslevel": 540, "pressure": 55, "latitude": 13.14876 }
```

.

**Bottom Screenshot:** Shows the same flow, but with the "IBM IoT" node selected and its properties configured. The "Properties" panel on the right shows the following settings:

- Authentication: API Key
- API Key: API KEY
- Input Type: Device Event
- Device Type: ☐ All or ☒ gasleakage114
- Device Id: ☐ All or ☒ device114
- Event: ☒ All or ☐ +
- Format: ☐ All or ☒ json
- QoS: 0
- Name: IBM IoT
- Service: registered

A note at the bottom of the properties panel states: "Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to".

# Develop A Web Application Using Node-RED Service.

The image displays two screenshots of the Node-RED web interface, illustrating the development of a web application using the Node-RED Service.

**Top Screenshot: Edit ibmiot node**

The interface shows the "Edit ibmiot node" configuration panel. The "Properties" section includes the following fields:

- Name: API KEY
- API Key: a-fytwic-ur5pzef8zj
- API Token: .....
- Server-Name: orgid.messaging.internetofthings.ibmcloud.com
- Scalable: ☐
- Application ID:
- Keep Alive: 60 Seconds
- Use Clean Session: ☒

The "debug" console on the right shows a log entry for the "Temperature" node, indicating a successful message payload.

**Bottom Screenshot: Edit function node**

The interface shows the "Edit function node" configuration panel. The "Properties" section includes the following fields:

- Name: Temperature

The "On Message" tab is selected, and the function code is as follows:

```
1 global.set('temperature',msg.payload.temperature)
2 msg.payload=msg.payload.temperature
3 return msg;
```

The "debug" console on the right shows a log entry for the "Temperature" node, indicating a successful message payload.

# Develop A Web Application Using Node-RED Service.

The image displays two screenshots of the Node-RED web interface, illustrating the development of a smart home dashboard.

**Top Screenshot: Edit function node**

- Flow 1:** A flow starting with an `IBM IoT` node, branching into `Temperature` and `Humidity` nodes, which then connect to a `msg` node. A `get data` node connects to a `webpage` node.
- Edit function node:** The node is named `Humidity`. The `On Message` tab is active, showing the following JavaScript code:

```
1 global.set("humidity",msg.payload.humidity)
2 msg.payload=msg.payload.humidity
3 return msg;
```
- Debug Console:** Shows a sequence of messages from the `2thtype/gasleakage114/d/device114/ev/status/rmq/json` endpoint. The messages include a `number` payload (37) and an `Object` payload containing temperature, gaslevel, pressure, and latitude data.

**Bottom Screenshot: Edit gauge node**

- Flow 1:** The flow is identical to the top screenshot.
- Edit gauge node:** The node is named `Temperature`. The `Properties tab is active, showing the following configuration:
  - Group: [Smart home] Group
  - Size: auto
  - Type: Gauge
  - Label: Temperature
  - Value format: {{value}}
  - Units: units
  - Range: min 0, max 60
  - Colour gradient: A gradient from green to red.
  - Sectors: 0, optional, optional, 60
  - Name: (empty)`
- Debug Console:** Shows the same sequence of messages as the top screenshot.

# Develop A Web Application Using Node-RED Service.

The image displays two screenshots of the Node-RED web interface, illustrating the development of a web application using the Node-RED Service.

**Top Screenshot: Edit gauge node**

The interface shows the "Edit gauge node" configuration panel. The node is named "Humidity" and is part of the "Smart home" Group. The Type is set to "Gauge". The Value format is set to "({value})". The Units are set to "units". The Range is set to "min 0 max 80". The Colour gradient is set to "0 ... optional ... optional ... 80". The Name is set to "Humidity". The node is enabled.

**Bottom Screenshot: Edit http in node**

The interface shows the "Edit http in node" configuration panel. The node is named "Name" and is part of the "Smart home" Group. The Method is set to "GET". The URL is set to "/data". The Name is set to "Name". The node is enabled.

Both screenshots show a flow diagram with nodes for "Temperature" and "Humidity" connected to a "msg" node, and a "get /data" node connected to a "webpage" node. The debug console on the right shows messages from the "get /data" node, including a JSON object with temperature, humidity, gas level, pressure, and latitude data.

# Develop A Web Application Using Node-RED Service.

The image displays two screenshots of the Node-RED web interface, illustrating the development of a web application.

**Top Screenshot: Edit function node**

- Name:** webpage
- Properties:** Setup, On Start, On Message, On Stop
- Code:**

```
1 msg.payload = { 'temperature': global.get('temperature'), 'humidity': global.get('humidity') }  
2 return msg;
```
- Debug Console:** Shows messages with payloads like `{ temperature: 34, humidity: 74, gaslevel: 540, pressure: 55, latitude: 13.14876 }`.

**Bottom Screenshot: Edit http response node**

- Name:** Name
- Status code:** msg.statusCode
- Headers:** (Empty list)
- Message:** The messages sent to this node must originate from an http input node
- Debug Console:** Shows messages with payloads like `{ temperature: 34, humidity: 74, gaslevel: 540, pressure: 55, latitude: 13.14876 }`.

Both screenshots show a flow diagram with nodes like `mqtt in`, `mqtt out`, `http in`, `http response`, `http request`, `websocket in`, `websocket out`, `tcp in`, and `tcp out`.