INDEX
SAN FRANCISCO

Discover. Collaborate. Deploy.

Talk to your Code

Michael Dawson

Please Note

- IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

About Michael Dawson

IBM Runtimes/IBM Node.js Community Lead

Node.js collaborator
Chair of Technical Steering Committee (TSC)
Community Committee member

Active in many working groups/teams

- Benchmarking
- Build
- Release
- N-API
- Diagnostics
- Security-wg
- User Feedback

Twitter: @mhdawson1
GitHub: @mhdawson
Linkedin: https://www.linkedin.com/in/michael-dawson-6051282

Overview

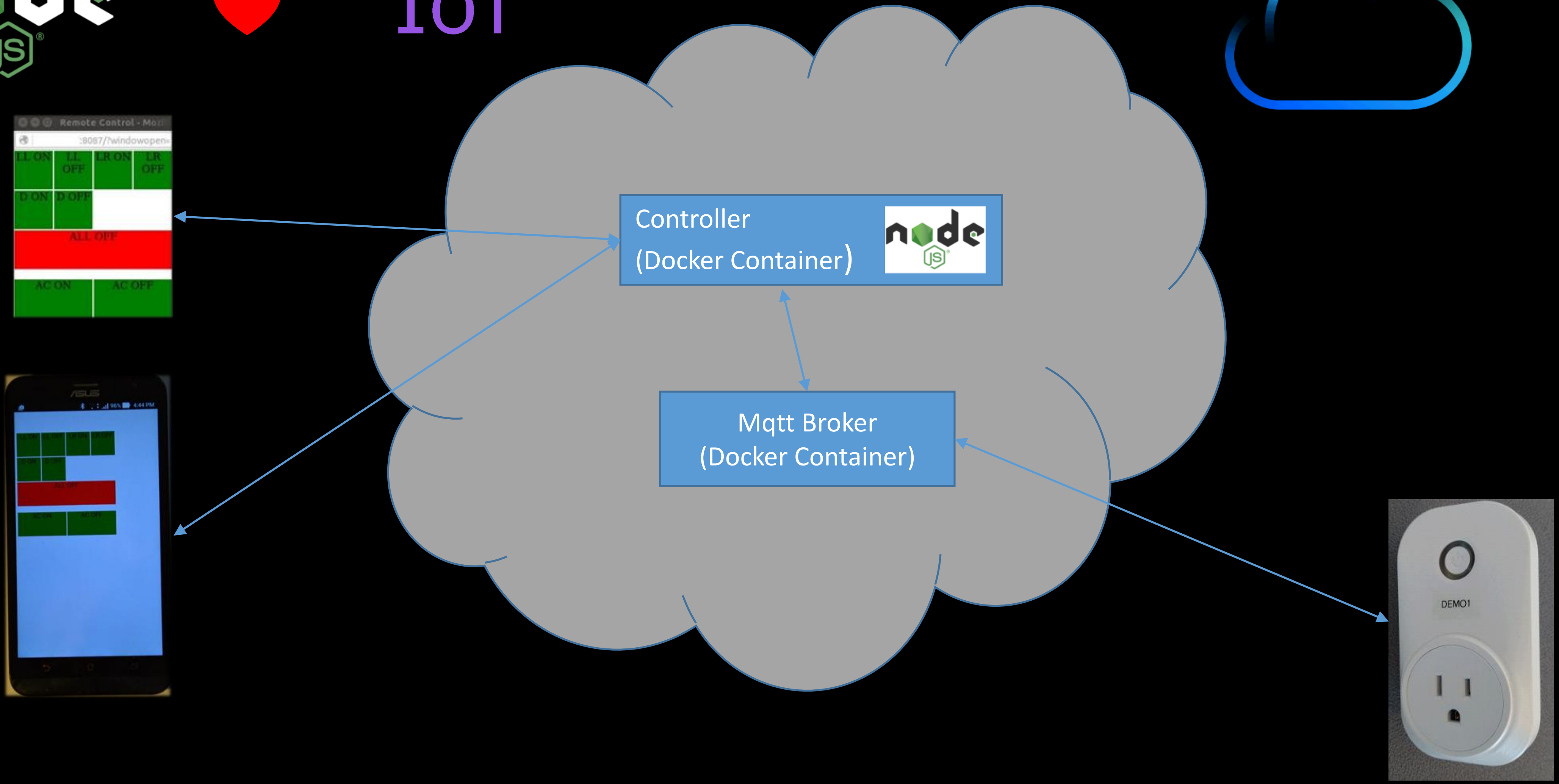
- node ❤ IoT
- Interaction V1
- Voice Service on the Rise
- Alexa to Mqtt Bridge
- Interaction V2
- Building an Alexa Skill
- Real world experience

node ♥ IoT

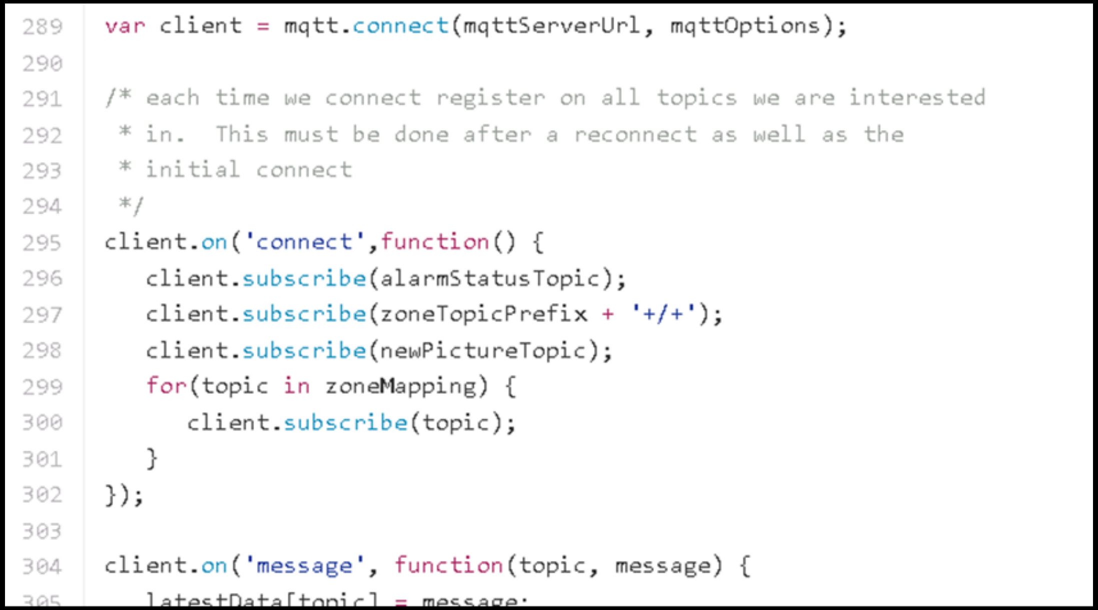
Async

- Internet of Things (IoT)
  - network of physically connected devices (things)
  - devices provide data
  - devices can be controlled
  - https://en.wikipedia.org/wiki/Internet_of_Things
- MQTT (MQ Telemetry Transport)
  - lightweight publish/subscribe
  - small footprint
  - low bandwidth (minimum size is 2 bytes)
  - From http://mqtt.org/ "MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol"
- node

IoT

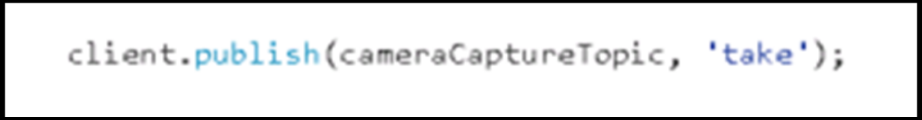

Remote Control - Mozi
:8087/?windowopen
LL ON
LL OFF
LR ON
LR OFF
D ON
D OFF
ALL OFF
AC ON
AC OFF
Controller
(Docker Container)
Mqtt Broker
(Docker Container)
DEMO1

IoT



```
var client = mqtt.connect(mqttServerUrl, mqttOptions);

/* each time we connect register on all topics we are interested
 * in.  This must be done after a reconnect as well as the
 * initial connect
 */
client.on('connect',function() {
   client.subscribe(alarmStatusTopic);
   client.subscribe(zoneTopicPrefix + '+/+');
   client.subscribe(newPictureTopic);
   for(topic in zoneMapping) {
      client.subscribe(topic);
   }
});

client.on('message', function(topic, message) {
   latestData[topic] = message;
```

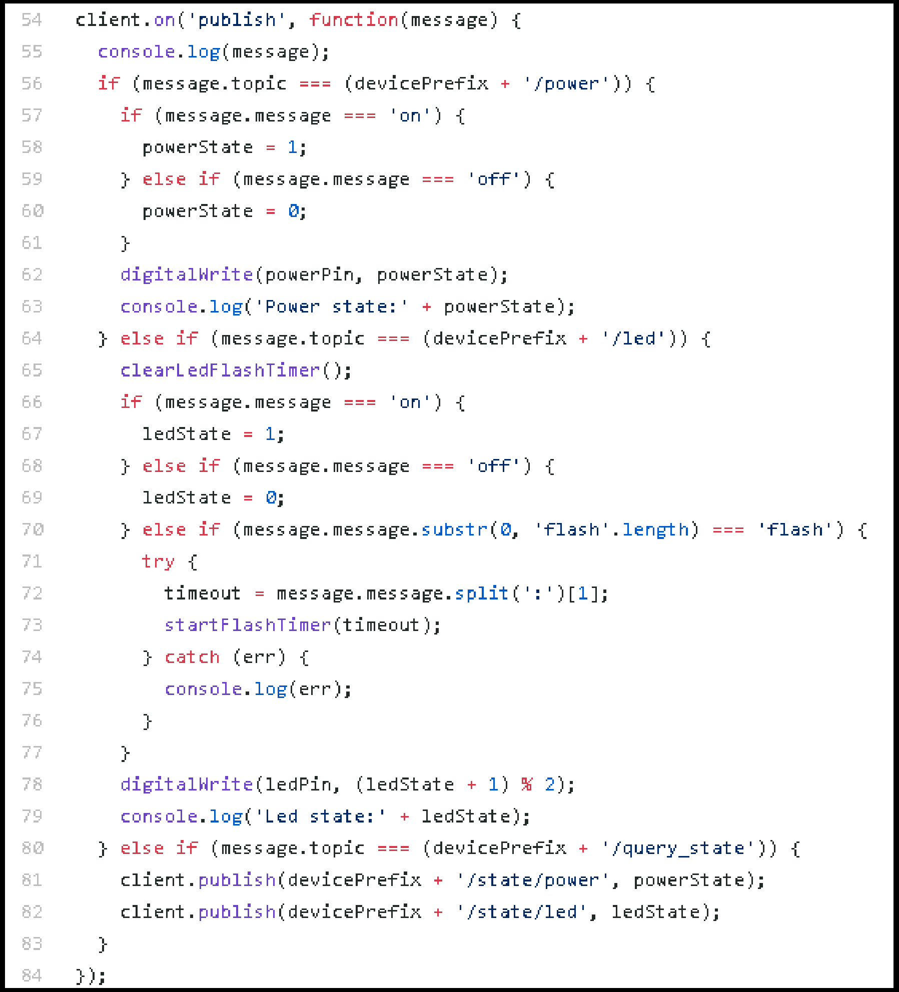
Receive



```
client.publish(cameraCaptureTopic, 'take');
```

Send

JavaScript ❤ IoT



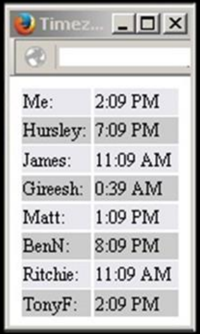
```
client.on('publish', function(message) {
  console.log(message);
  if (message.topic === (devicePrefix + '/power')) {
    if (message.message === 'on') {
      powerState = 1;
    } else if (message.message === 'off') {
      powerState = 0;
    }
    digitalWrite(powerPin, powerState);
    console.log('Power state:' + powerState);
  } else if (message.topic === (devicePrefix + '/led')) {
    clearLedFlashTimer();
    if (message.message === 'on') {
      ledState = 1;
    } else if (message.message === 'off') {
      ledState = 0;
    } else if (message.message.substr(0, 'flash'.length) === 'flash') {
      try {
        timeout = message.message.split(':')[1];
        startFlashTimer(timeout);
      } catch (err) {
        console.log(err);
      }
    }
    digitalWrite(ledPin, (ledState + 1) % 2);
    console.log('Led state:' + ledState);
  } else if (message.topic === (devicePrefix + '/query_state')) {
    client.publish(devicePrefix + '/state/power', powerState);
    client.publish(devicePrefix + '/state/led', ledState);
  }
});
```

https://github.com/mhdawson/espruino-stuff/blob/master/SmartPlug.js

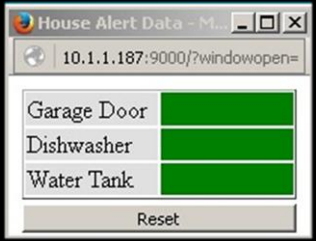
Interaction – V1

- Demo

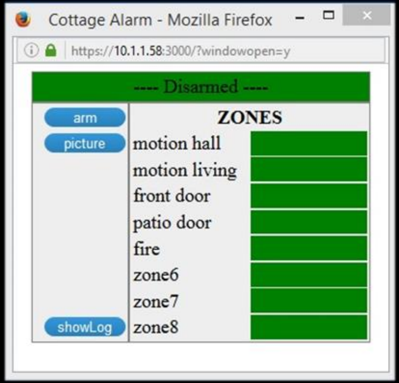
Good enough, create bunch of Apps



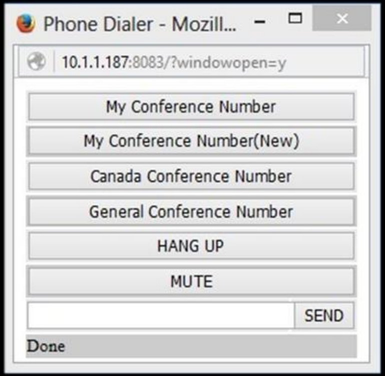
Me:	2:09 PM
Hursley:	7:09 PM
James:	11:09 AM
Gireesh:	0:39 AM
Matt:	1:09 PM
BenN:	8:09 PM
Ritchie:	11:09 AM
TonyF:	2:09 PM



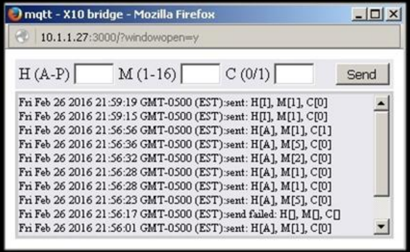
Garage Door	🟩
Dishwasher	🟩
Water Tank	🟩


Cottage Alarm - Mozilla Firefox
https://10.1.1.58:3000/?windowopen=y
---- Disarmed ----
arm
picture
showLog

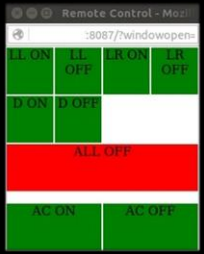
ZONES	
motion hall	🟩
motion living	🟩
front door	🟩
patio door	🟩
fire	🟩
zone6	🟩
zone7	🟩
zone8	🟩

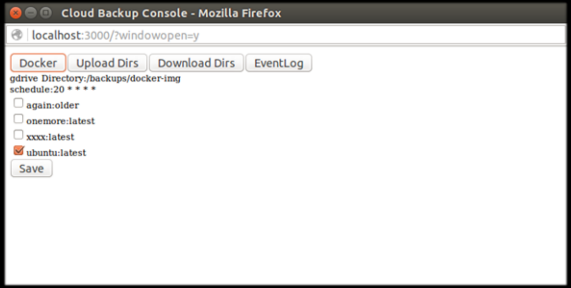


825782
15

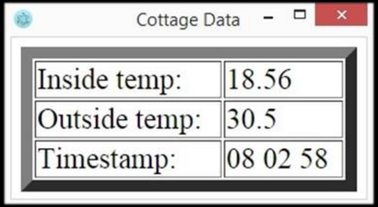

mqtt - X10 bridge - Mozilla Firefox
10.1.1.27:3000/?windowopen=y
H (A-P)
M (1-16)
C (0/1)
Send

```
Fri Feb 26 2016 21:59:19 GMT-0500 (EST):sent: H[I], M[1], C[0]
Fri Feb 26 2016 21:59:15 GMT-0500 (EST):sent: H[I], M[1], C[0]
Fri Feb 26 2016 21:56:56 GMT-0500 (EST):sent: H[A], M[1], C[1]
Fri Feb 26 2016 21:56:36 GMT-0500 (EST):sent: H[A], M[5], C[0]
Fri Feb 26 2016 21:56:32 GMT-0500 (EST):sent: H[A], M[2], C[0]
Fri Feb 26 2016 21:56:28 GMT-0500 (EST):sent: H[A], M[1], C[0]
Fri Feb 26 2016 21:56:28 GMT-0500 (EST):sent: H[A], M[1], C[0]
Fri Feb 26 2016 21:56:23 GMT-0500 (EST):sent: H[A], M[5], C[0]
Fri Feb 26 2016 21:56:17 GMT-0500 (EST):send failed: H[], M[], C[]
Fri Feb 26 2016 21:56:01 GMT-0500 (EST):sent: H[A], M[1], C[0]
```

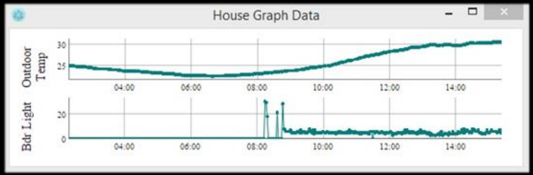



Cloud Backup Console - Mozilla Firefox
localhost:3000/?windowopen=y
Docker
Upload Dirs
Download Dirs
EventLog
Save

gdrive Directory:/backups/docker-img
schedule:20 * * * *
☐ again:older
☐ onemore:latest
☐ xxxx:latest
☑ ubuntu:latest

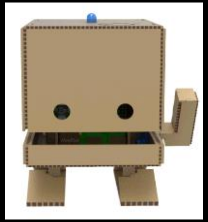


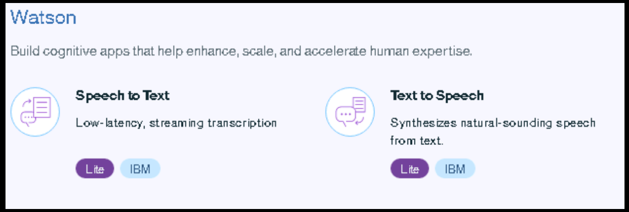
Inside temp:	18.56
Outside temp:	30.5
Timestamp:	08 02 58



Voice services on the Rise

- IBM Watson





Watson

Build cognitive apps that help enhance, scale, and accelerate human expertise.

Speech to Text

Low-latency, streaming transcription

Lite IBM

Text to Speech

Synthesizes natural-sounding speech from text.

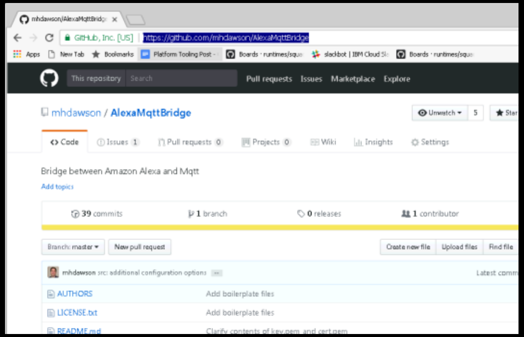
Lite IBM

https://www.ibm.com/watson/services/speech-to-text/

- Google Home

- Amazon Alexa

Alexa to Mqtt Bridge



mhdawson / AlexaMqttBridge

Bridge between Amazon Alexa and Mqtt

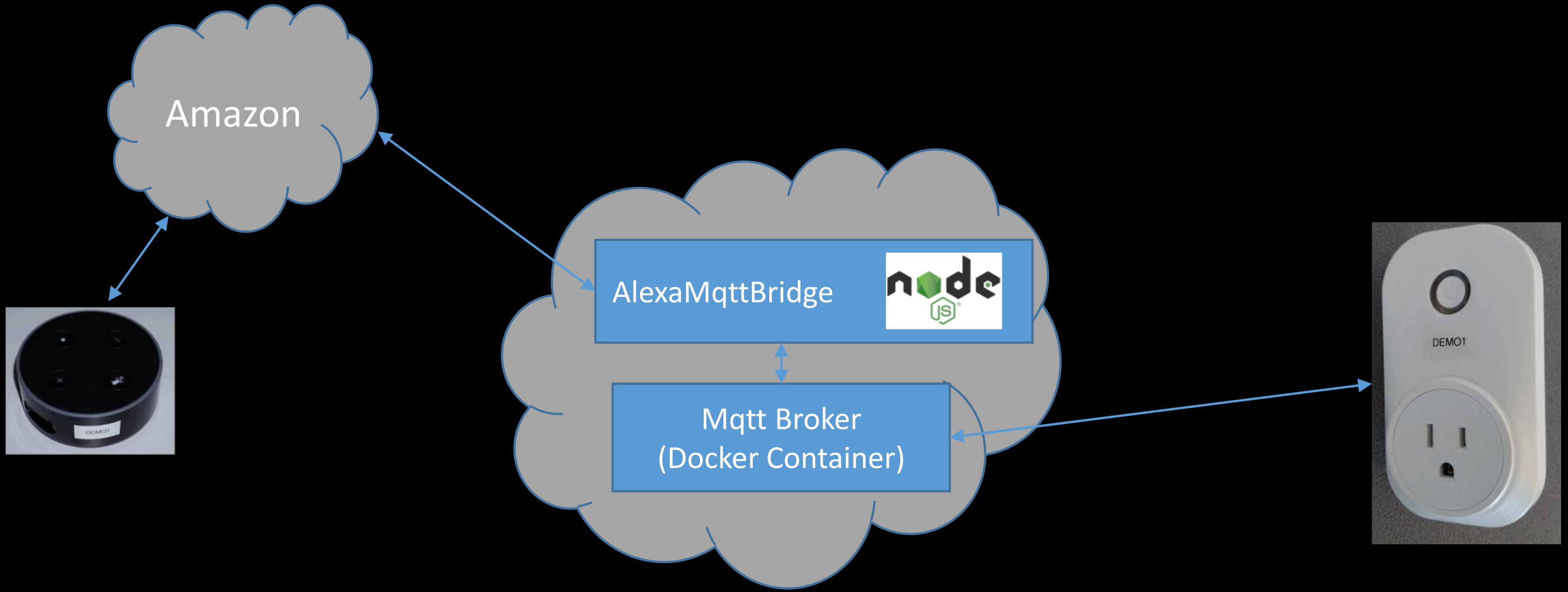
Add topics

39 commits | 1 branch | 0 releases | 1 contributor

mhdawson src: additional configuration options

AUTHORS	Add boilerplate files
LICENSE.txt	Add boilerplate files
README.md	Clarify contents of key.pem and cert.pem

- Convert Voice Request to Mqtt Message (and vice-versa)
- https://github.com/mhdawson/AlexaMqttBridge



Interaction V2

- Demo

Alexa to Mqtt Bridge

- Server in
- Alexa Skill

Writing an Alexa Skill

- Skill Information
- Interaction Model
  - Intent
  - Slots/Slot Types
  - Sample Utterances
- Endpoint

Skill Information

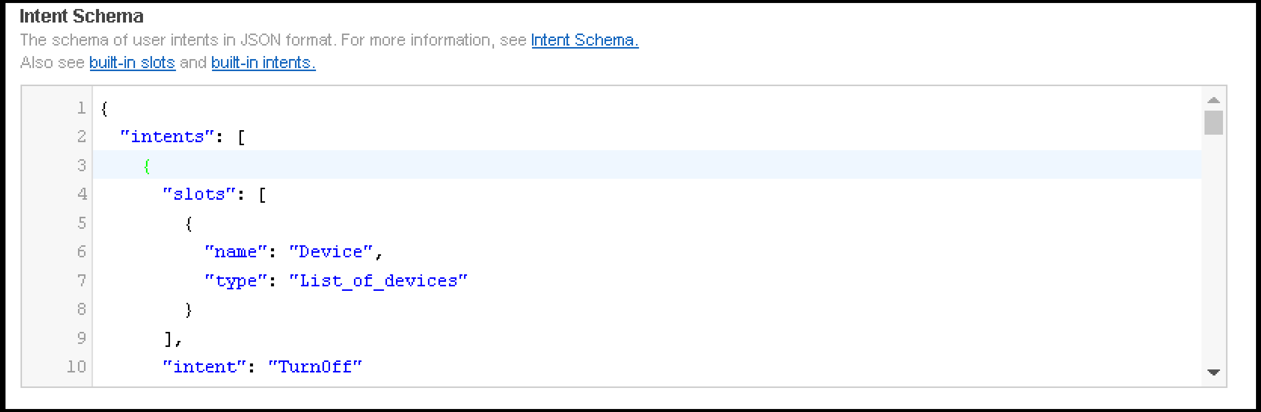
- Name
- Invocation Name

Ask **Michael** to …

- Language (new, at least to me ☺)

Intents

```
{
 "intents": [
  {
   "slots": [
    {
     "name": "Device",
     "type": "List_of_devices"
    }
   ],
   "intent": "TurnOff"
  },
  {
   "slots": [
    {
     "name": "Device",
     "type": "List_of_devices"
    }
   ],
   "intent": "TurnOn"
  },
…
```


Intent Schema

The schema of user intents in JSON format. For more information, see Intent Schema.
Also see built-in slots and built-in intents.

```
{
  "intents": [
    {
      "slots": [
        {
          "name": "Device",
          "type": "List_of_devices"
        }
      ],
      "intent": "TurnOff"
```

Slot Types

- List of options for Slot
  - Does not have to be exhaustive
  - Helps recognition
- Built in types as well
  - AMAZON.TVSeries, AMAZON.NUMBER
- Custom example:
  - List_of_devices

living room lights
living room right
living room left
dining room
power monitor
TV
TV 1
alarm
andrew
office light
fan
fan low
fan hi
fan medium
media1
Tv1
Socket
socketlight

Sample Utterances

- Samples of expected speech patterns

```
TurnOff Turn off {Device}
TurnOff to Turn off {Device}
TurnOn Turn on {Device}
TurnOn to Turn on {Device}
Tune {Device} to channel {Channel}
Tune {Device} channel {Channel}
Tune set {Device} channel {Channel}
Tune set {Device} to channel {Channel}
Tune to set {Device} channel {Channel}
Tune to set {Device} to channel {Channel}
VolumeUp {Device} volume up {Repeat}
VolumeUp to turn {Device} volume up {Repeat}
VolumeUp turn {Device} volume up {Repeat}
VolumeDown {Device} volume down {Repeat}
VolumeDown to turn {Device} volume down {Repeat}
VolumeDown turn {Device} volume down {Repeat}
Mute mute {Device}
Pause pause {Device}
UnPause unpause {Device}
Stop stop {Device}
Seek seek {Device} to {Time}
```

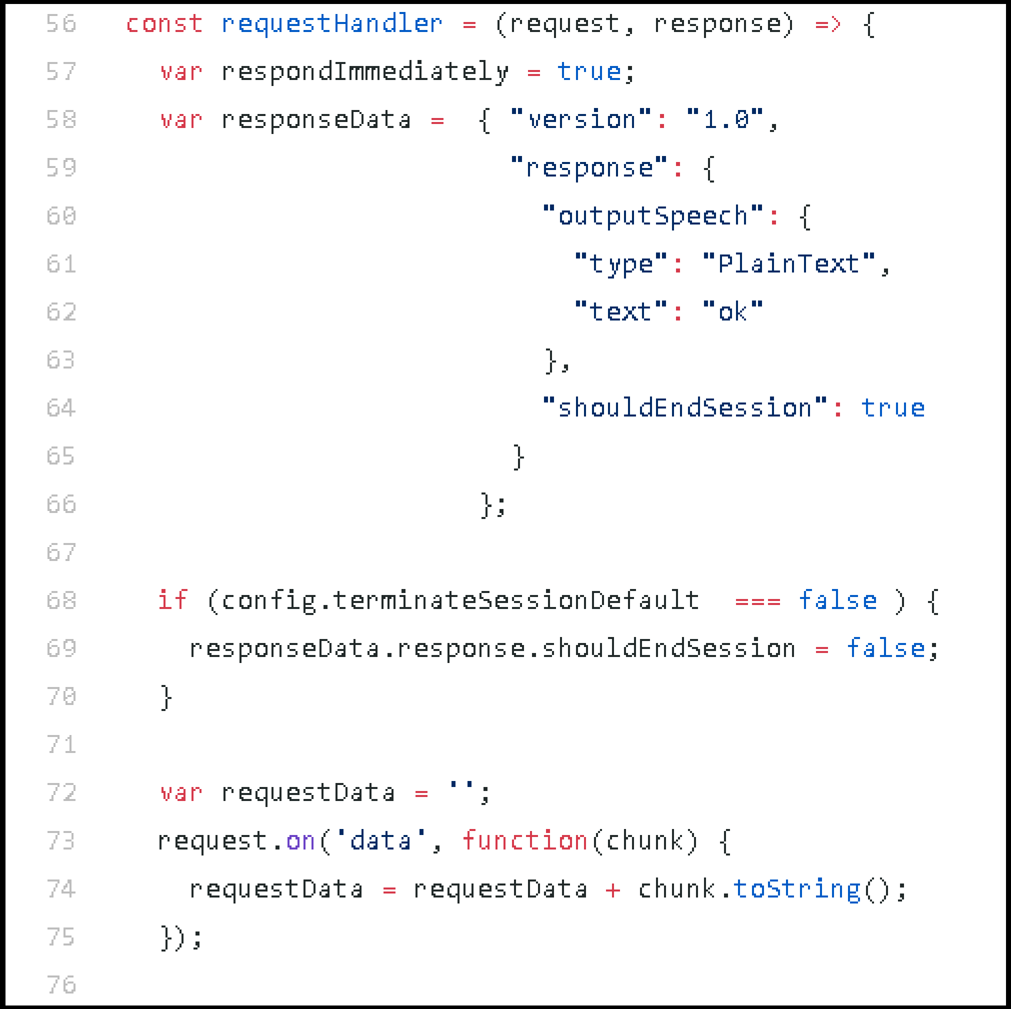
Endpoint

- AWS Lamda
- Microservice
  - https://alexademo.devrus.com/alexa?XXXXXXX
  - Must be SSL
    - Trusted cert authority
      - Let's Encrypt™ certificate - https://letsencrypt.org/
      - CloudFlare
      - Cloud provider (ex IBM Cloud)
    - Upload self-signed

Server Config

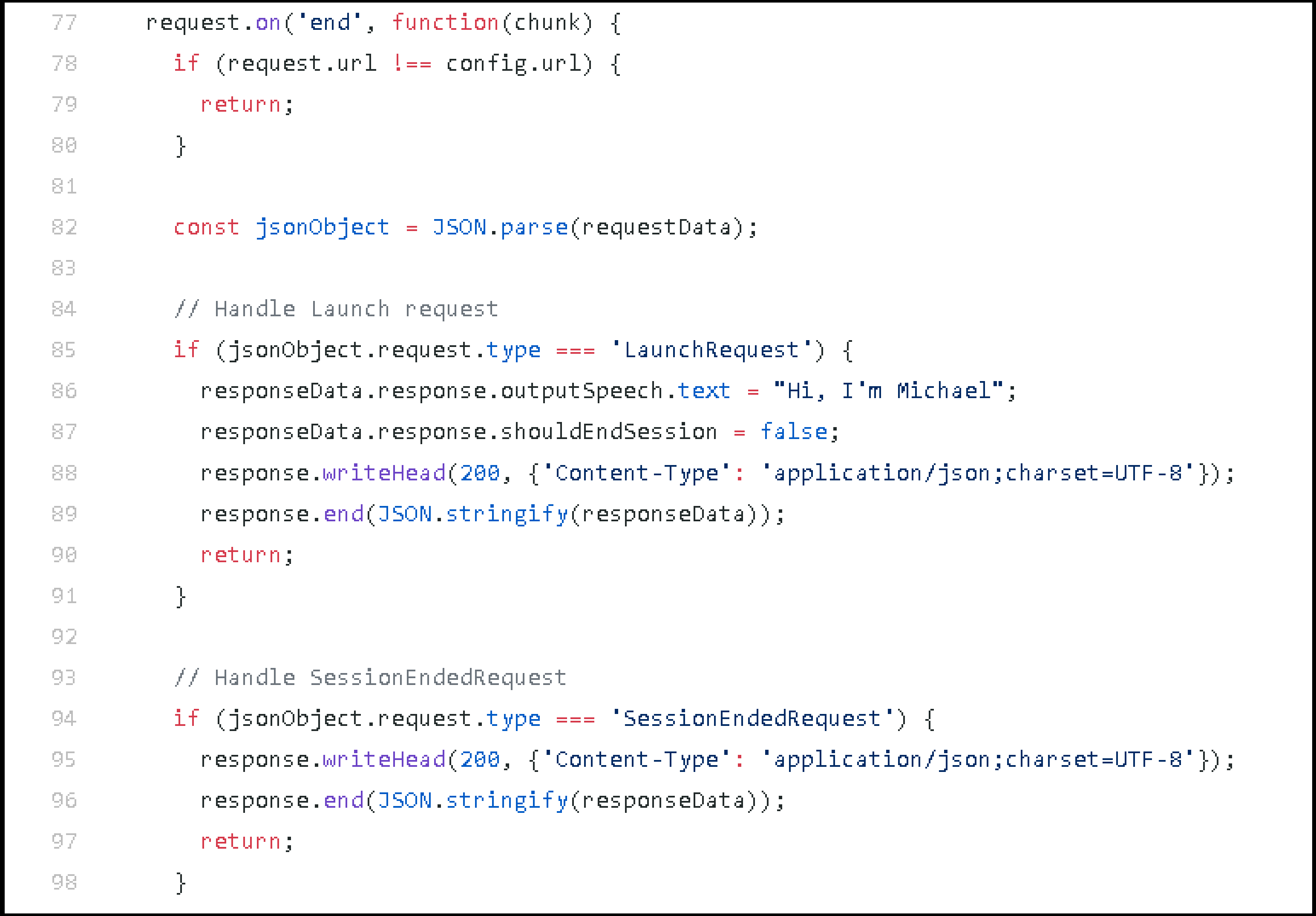
```
{
  "logging": false,
  "port": 5000,
  "terminateSessionDefault": true,
  "url": "/alexa?XXXXXXXXXXXXXXXXX",
  "intents": { "TurnOn": { "alarm": { "topic": "house/alarm/control", "message": "arm" },
                           "livingroomright": { "topic": "house/x10", "message": "A,1,1" },
                           "diningroom": { "topic": "house/x10", "message": "A,5,1" },
                           "officelight": { "topic": "home/2272/200", "message": "0F0FFFFF0101" },
                           "ac": { "topic": "home/2272", "message": "0F0FFFFF0110" },
                           "socket": { "topic": "house/esp2/power", "message": "on" },
                           "socketlight": { "topic": "house/esp2/led", "message": "on" }
                         },
               "TurnOff": { "livingroomlights": [ { "topic": "house/x10", "message": "A,5,0" },
                                                  { "topic": "house/x10", "message": "A,1,0:1000" },
                                                  { "topic": "house/x10", "message": "A,2,0:2000" } ],
                            "livingroomright": { "topic": "house/x10", "message": "A,1,0" },
                            "diningroom": { "topic": "house/x10", "message": "A,5,0" },
                            "officelight": { "topic": "home/2272/200", "message": "0F0FFFFF0110" },
                            "socket": { "topic": "house/esp2/power", "message": "off" },
                            "socketlight": { "topic": "house/esp2/led", "message": "off" }
                          },

               "WhatsNew": { "tv": { "topic": "house/dlnaplay/control",
                                     "message": "whatsnew",
                                     "responseTopic": "house/dlnaplay/response" } },

               "Recipe": { "default": { "topic": "pdfViewer/request",
                                        "message": "${slots.RecipeNames.value}" } },

             },
  "mqtt": { "serverUrl": "mqtt:XXXX:1883"
          },
  "mqttExternal": { "serverUrl": "mqtts:XXXXX:8883"
          }
}
```

Server



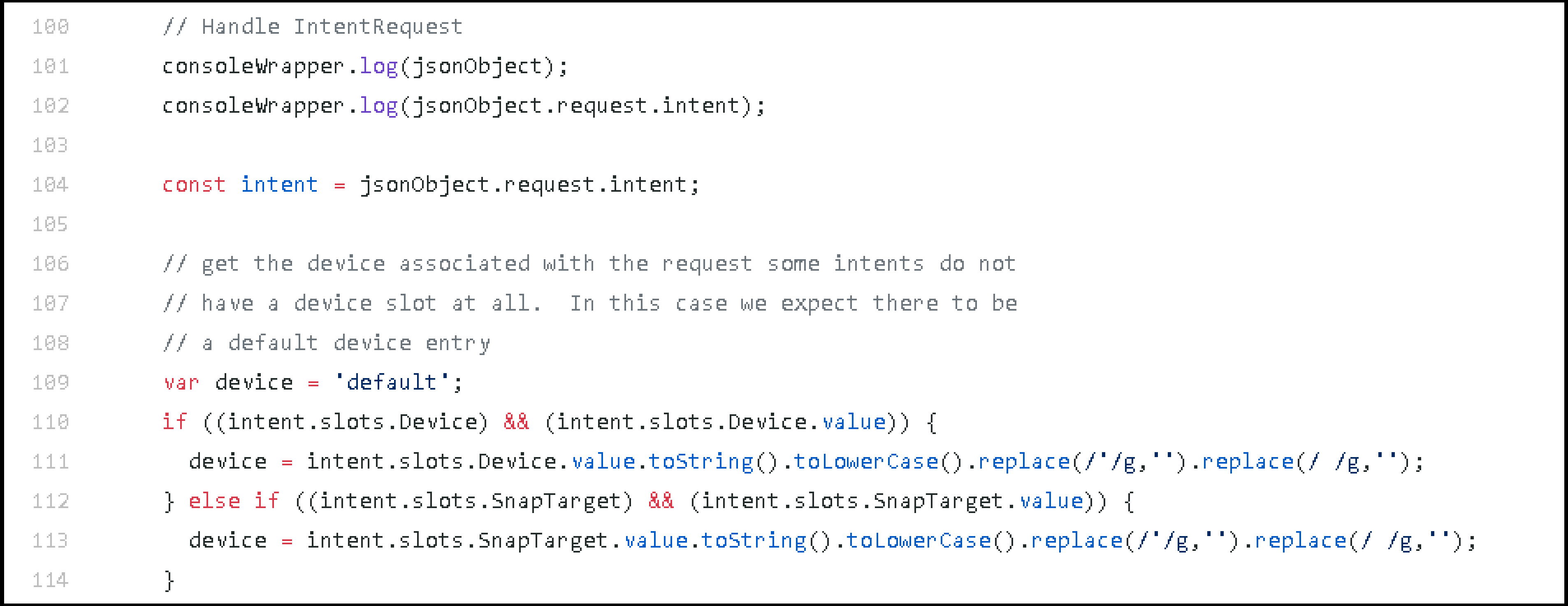
```javascript
const requestHandler = (request, response) => {
  var respondImmediately = true;
  var responseData =  { "version": "1.0",
                        "response": {
                          "outputSpeech": {
                            "type": "PlainText",
                            "text": "ok"
                          },
                          "shouldEndSession": true
                        }
                      };

  if (config.terminateSessionDefault  === false ) {
    responseData.response.shouldEndSession = false;
  }

  var requestData = '';
  request.on('data', function(chunk) {
    requestData = requestData + chunk.toString();
  });

```

Server



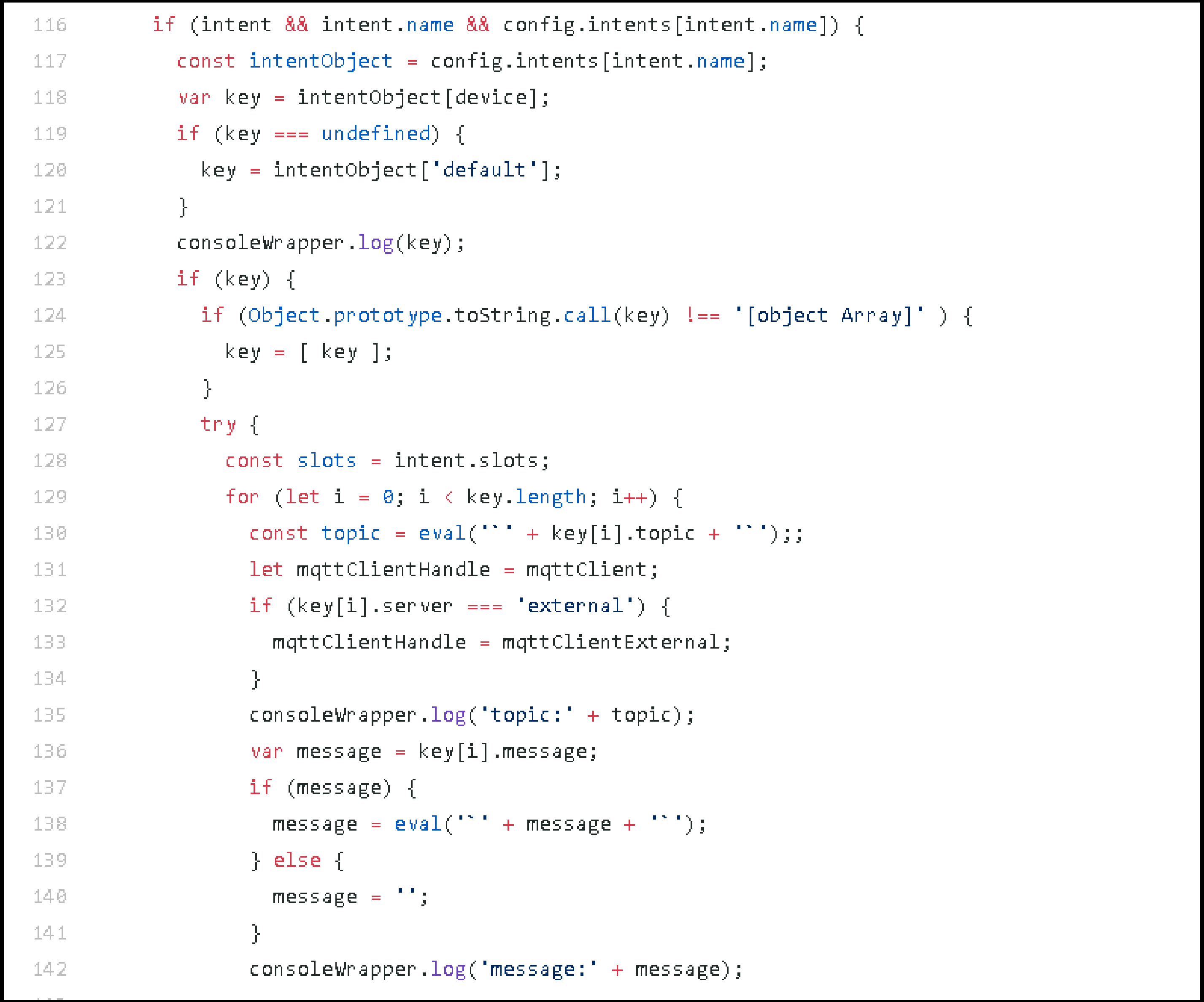
```
  request.on('end', function(chunk) {
    if (request.url !== config.url) {
      return;
    }

    const jsonObject = JSON.parse(requestData);

    // Handle Launch request
    if (jsonObject.request.type === 'LaunchRequest') {
      responseData.response.outputSpeech.text = "Hi, I'm Michael";
      responseData.response.shouldEndSession = false;
      response.writeHead(200, {'Content-Type': 'application/json;charset=UTF-8'});
      response.end(JSON.stringify(responseData));
      return;
    }

    // Handle SessionEndedRequest
    if (jsonObject.request.type === 'SessionEndedRequest') {
      response.writeHead(200, {'Content-Type': 'application/json;charset=UTF-8'});
      response.end(JSON.stringify(responseData));
      return;
    }
```

Server



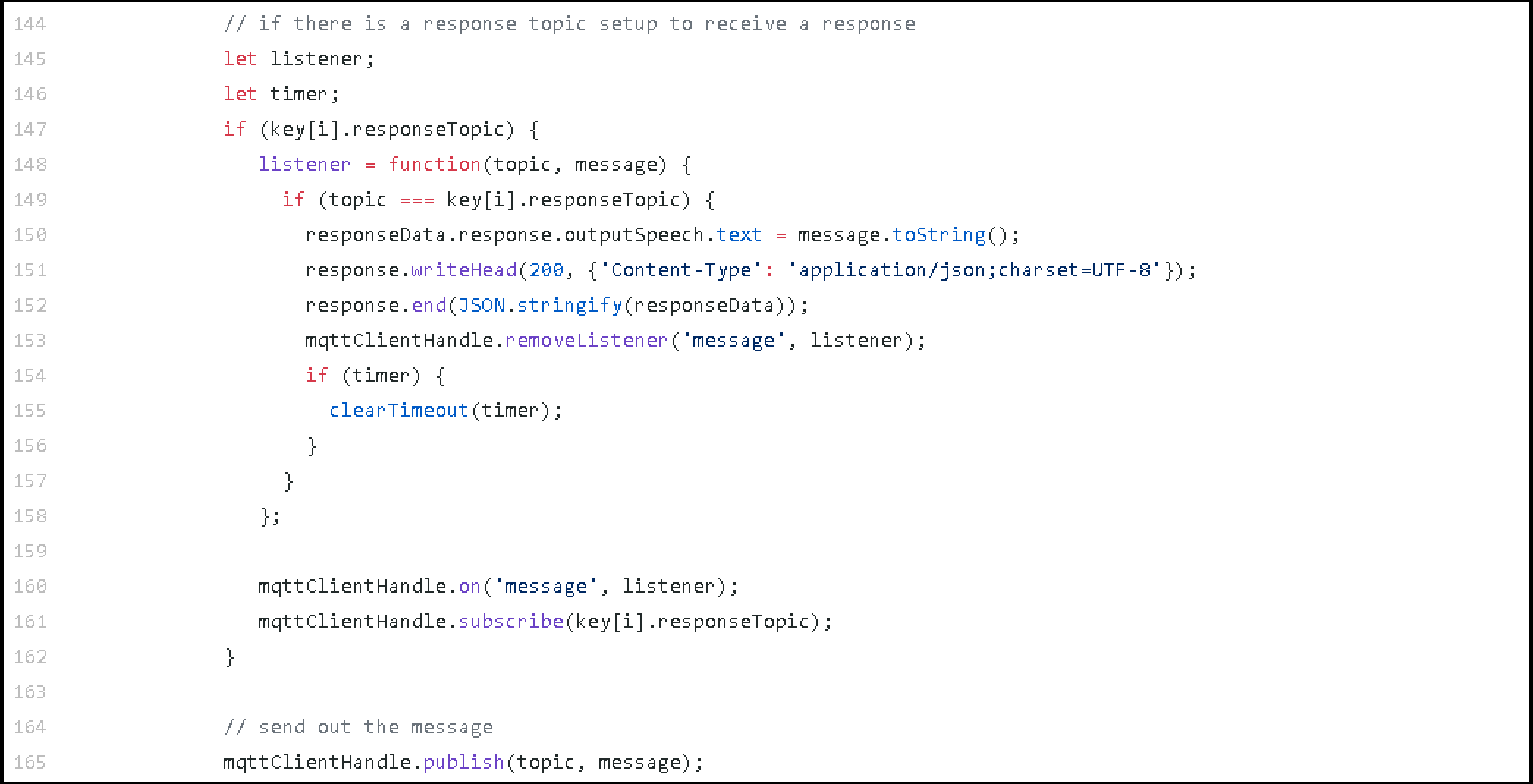
```javascript
// Handle IntentRequest
consoleWrapper.log(jsonObject);
consoleWrapper.log(jsonObject.request.intent);

const intent = jsonObject.request.intent;

// get the device associated with the request some intents do not
// have a device slot at all.  In this case we expect there to be
// a default device entry
var device = 'default';
if ((intent.slots.Device) && (intent.slots.Device.value)) {
  device = intent.slots.Device.value.toString().toLowerCase().replace(/'/g,'').replace(/ /g,'');
} else if ((intent.slots.SnapTarget) && (intent.slots.SnapTarget.value)) {
  device = intent.slots.SnapTarget.value.toString().toLowerCase().replace(/'/g,'').replace(/ /g,'');
}
```

Server



```
    if (intent && intent.name && config.intents[intent.name]) {
      const intentObject = config.intents[intent.name];
      var key = intentObject[device];
      if (key === undefined) {
        key = intentObject['default'];
      }
      consoleWrapper.log(key);
      if (key) {
        if (Object.prototype.toString.call(key) !== '[object Array]' ) {
          key = [ key ];
        }
        try {
          const slots = intent.slots;
          for (let i = 0; i < key.length; i++) {
            const topic = eval('`' + key[i].topic + '`');;
            let mqttClientHandle = mqttClient;
            if (key[i].server === 'external') {
              mqttClientHandle = mqttClientExternal;
            }
            consoleWrapper.log('topic:' + topic);
            var message = key[i].message;
            if (message) {
              message = eval('`' + message + '`');
            } else {
              message = '';
            }
            consoleWrapper.log('message:' + message);
```

Server



```javascript
// if there is a response topic setup to receive a response
let listener;
let timer;
if (key[i].responseTopic) {
  listener = function(topic, message) {
    if (topic === key[i].responseTopic) {
      responseData.response.outputSpeech.text = message.toString();
      response.writeHead(200, {'Content-Type': 'application/json;charset=UTF-8'});
      response.end(JSON.stringify(responseData));
      mqttClientHandle.removeListener('message', listener);
      if (timer) {
        clearTimeout(timer);
      }
    }
  };

  mqttClientHandle.on('message', listener);
  mqttClientHandle.subscribe(key[i].responseTopic);
}

// send out the message
mqttClientHandle.publish(topic, message);
```

Demo

- Show debug/console output

What Works?

- Search
  - Play next episode
  - Recipes
- Hands free

What Doesn’t

- Repeated actions
  - Volume up/down
- Noisy environments
- When you demo to your friends/family ☺

Summary and Questions

Notices and disclaimers

Notices and disclaimers continued

INDEX
SAN FRANCISCO

Discover. Collaborate. Deploy.

Talk to your Code

Michael Dawson