

Practicum 1 - Cloud & Security

1. Run nogmaals opgave1.js. Waar start dit node.js programma? Er is geen main functie. Wanneer stopt het node.js programma en waarom is dit zo?
 - Dit programma start boven aan het bestand zolang er code buiten een functie blok staat. Dit programma stopt wanneer alle functies/ code is uitgevoerd.
2. Voeg een timer toe die de print functie aanroept. Wanneer stopt het node.js script nu?
 - het script stopt nadat de callback van de timer is uitgevoerd
3. Voeg nu een setInterval toe. Wanneer stopt het programma nu?
 - Het programma stopt wanneer de computer crasht of iemand het programma handmatig uitzet
4. Voeg nu een timer toe die het interval weer cancelt na 20 seconde? Wanneer stopt nu het script?
 - Het programma stopt na 20 seconden
5. Het inlezen van keyboard input kan met de volgende code:
<https://nodejs.org/docs/latest/api/readline.html>. Test deze code

```
const readline = require('node:readline');
const { stdin: input, stdout: output } = require('node:process');

const rl = readline.createInterface({ input, output });

rl.question('What do you think of Node.js? ', (answer) => {
  // TODO: Log the answer in a database
  console.log(`Thank you for your valuable feedback: ${answer}`);

  rl.close();
});
```

```
What do you think of Node.js?
Thank you for your valuable feedback: very cool
```

6. Waar zit in deze code een anonieme functie

Het volgende stuk code is een anonieme functie: `` (answer) => { console.log(Thank you for your valuable feedback: \${answer});

rl.close(); }); ``

7. Herschrijf de code naar een niet anonieme callback functie

```
const readline = require('node:readline');
const { stdin: input, stdout: output } = require('node:process');

const rl = readline.createInterface({ input, output });

function handleInput(answer) {
  console.log(`Thank you for your valuable feedback: ${answer}`);
  rl.close();
}

rl.question('What do you think of Node.js? ', handleInput);
```

8. Blokkeert de question aanroep de event loop? Test dit mbv een timer.

- Nee, dit blokkeert niet de event loop

9. Optioneel: Pas de code zo aan dat je een tweede vervolg vraag stelt

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// very scalable, can even ask more than two questions :)
const questions = ["give answer 1: ", "give answer 2: "];

let index = 0;

// ask question and read in input
function askQuestion() {
  if (index >= questions.length) {
    rl.close();
    process.exit(0);
  }

  rl.question(questions[index], handleInput);
  index++;
}

function handleInput(input) {
  console.log(`Received input: ${input}`);
  askQuestion();
}

// start code
askQuestion();
```

```
give answer 1: antwoord 1
Received input: antwoord 1
give answer 2: antwoord 2
Received input: antwoord 2
```

10. Gebruik de event emitter om meerdere keren invoer van het toetsenbord te lezen. (Event: 'line')

```
const readline = require('node:readline');
const { stdin: input, stdout: output } = require('node:process');

const rl = readline.createInterface({ input, output });

function handleInput(answer) {
  console.log(`Thank you for your valuable feedback: ${answer}`);
}

rl.on('line', handleInput);
```

```
a
Thank you for your valuable feedback: a
a
Thank you for your valuable feedback: a
a
Thank you for your valuable feedback: a
a
Thank you for your valuable feedback: a
a
Thank you for your valuable feedback: a
a
Thank you for your valuable feedback: a
a
Thank you for your valuable feedback: a
```

11. Laat het programma stoppen als er quit wordt ingetypt

```
const readline = require('node:readline');
const { stdin: input, stdout: output } = require('node:process');

const rl = readline.createInterface({ input, output });

function handleInput(answer) {
  if (answer == "quit") {
    rl.close();
  }
}
```

```
    console.log(`Thank you for your valuable feedback: ${answer}`);  
  }  
  
  rl.on('line', handleInput);
```

```
a  
Thank you for your valuable feedback: a  
quit  
Thank you for your valuable feedback: quit  
PS C:\Users\iwanv\Documents\school\IoT\Cloud\opdracht1> █
```

12. Maak een simpel programma dat een mqtt bericht published mbv het mqtt package. Test of het bericht is aangekomen via de webpagina van hivemq.

```
const mqtt = require('mqtt');  
const brokerUrl = 'mqtt://broker.hivemq.com';  
const topic = 'mqttiottest87777';  
const message = 'Hello, MQTT!';  
const client = mqtt.connect(brokerUrl);  
  
client.on('connect', () => {  
  console.log('Connected to MQTT broker');  
  
  client.publish(topic, message, (error) => {  
    if (error) {  
      console.error('Failed to publish message:', error);  
    } else {  
      console.log('Message published successfully');  
    }  
    client.end();  
  });  
});
```

Messages



2024-09-02 17:30:06	Topic: mqttiottest87777	Qos: 0
Hello, MQTT!		

2024-09-02 17:29:29	Topic: mqttiottest87777	Qos: 0
Hello, MQTT!		

13. Maak een andere simpel programma dat een mqtt bericht ontvangt door te subscriben mbv het mqtt package. Publiken van een bericht kan zowel via het programma van de vorige vraag als via de webpagina van hivemq

```
const mqtt = require('mqtt');
const brokerUrl = 'mqtt://broker.hivemq.com';
const topic = 'mqttiottest87777';
const client = mqtt.connect(brokerUrl);

client.on('connect', () => {
  console.log('Connected to MQTT broker');

  client.subscribe(topic, (error) => {
    if (error) {
      console.error('Failed to subscribe to topic:', error);
    } else {
      console.log('Subscribed succesfully');
    }
  });
});

client.on('message', (mtopic, message) => {
  console.log(`Received message on topic ${mtopic}: ${message}`);
});
```

Publish



Topic

QoS

Retain



Publish

Message

Connected to MQTT broker
Subscribed succesfully
Received message on topic mqttiottest87777: asdsad

14. Demonstreer nu dat 1 publish bij meerdere subscribers terecht kan komen. Hiermee demonstreer je het publish subscribe pattern

```
Connected to MQTT broker
Subscribed succesfully
Received message on topic mqttiottest87777: Very cool message
```

Messages

2024-09-02 17:39:34 Topic: mqttiottest87777 Qos: 0
Very cool message

15. Maak een simulated bewegingssensor waarbij het intypen van een toets betekent dat er beweging is geconstateerd. Er wordt dan een JSON bericht via mqtt verstuurd

```
const mqtt = require('mqtt');
const readline = require('readline');

const brokerUrl = 'mqtt://broker.hivemq.com';
const topic = 'mqttBewegingIoTHHS';
const client = mqtt.connect(brokerUrl);

client.on('connect', () => {
  console.log('Connected to MQTT broker');

  client.publish(topic, 'connection established', (error) => {
    if (error) {
      console.error('Failed to publish message:', error);
    } else {
      console.log('Message published successfully');
    }
  });
});

// Set up readline to listen for key presses
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.on('line', (input) => {

  client.publish(topic, JSON.stringify({'Movement found on': input}),
(error) => {
  if (error) {
    console.error('Failed to publish message:', error);
  } else {
    console.log('Message published successfully');
  }
});
});
});
```

Messages



2024-09-02 17:50:57 Topic: mqttBewegingIoTHHS Qos: 0
{"Movement found on":"a"}

2024-09-02 17:50:55 Topic: mqttBewegingIoTHHS Qos: 0
connection established

16. Maak een simulated lot lamp (actuator) die aangaat als er beweging is geconstateerd. Via de ontvangen JSON bericht kan de lamp zowel aan met een bepaalde lichtsterkte en ook helemaal uit. Simuleer de lamp door iets te printen.

```
const mqtt = require('mqtt');
const brokerUrl = 'mqtt://broker.hivemq.com';
const topic = 'mqttBewegingIoTHHS';
const client = mqtt.connect(brokerUrl);

let lampAan = false;
let strengthLevel = 0;

client.on('connect', () => {
  console.log('Connected to MQTT broker');

  client.subscribe(topic, (error) => {
    if (error) {
      console.error('Failed to subscribe to topic:', error);
    } else {
      console.log('Subscribed succesfully');
    }
  });
});

function containsValue(obj, value) {
  for (const key in obj) {
    if (obj[key] === value) {
      return true;
    }
  }
  return false;
}

// 'a' turns lamp on or off
// 'b' controls strength
client.on('message', (mtopic, message) => {

  const jsonMessage = JSON.parse(message.toString());
  // console.log(`Received message on topic ${mtopic}:`, jsonMessage);
```

```

    if (containsValue(jsonMessage, 'a')) {
      if (lampAan) {
        lampAan = false;
        strengthLevel = 0;
        console.log("lamp turns off")
      }
      else {
        lampAan = true;
        console.log("lamp turns on")
      }
    }
    else if (containsValue(jsonMessage, 'b') && lampAan) {
      strengthLevel++;
      if (strengthLevel > 5)
        strengthLevel = 0;
      console.log('Light strength: ' + strengthLevel)
    }
  });

```

```

Connected to MQTT broker
Subscribed successfully
lamp turns on
light strength: 1
light strength: 2
lamp turns off

```

```

Connected to MQTT broker
Message published successfully
a
Message published successfully
b
Message published successfully
b
Message published successfully
a
Message published successfully

```

17. Zorg dat de lamp automatisch uitgaat na 5 seconden. Maak 2 varianten. Eén variant waarbij de logica (voor het automatisch uitgaan) op de sensor zit en één variant waar de logica op de actuator zit.

Automatisch uitgaan actuator:

```

const mqtt = require('mqtt');
const brokerUrl = 'mqtt://broker.hivemq.com';
const topic = 'mqttBewegingIoTHHS';
const client = mqtt.connect(brokerUrl);

let lampEnabled = false;
let strengthLevel = 0;
let timeoutId;

client.on('connect', () => {
  console.log('Connected to MQTT broker');

  client.subscribe(topic, (error) => {
    if (error) {
      console.error('Failed to subscribe to topic:', error);
    } else {
      console.log('Subscribed successfully');
    }
  });
});

function containsValue(obj, value) {

```



```

    for (const key in obj) {
        if (obj[key] === value) {
            return true;
        }
    }
    return false;
}

function controllLamp(lampBool) {
    if (lampBool) {
        lampEnabled = false;
        strengthLevel = 0;
        console.log("lamp turns off")
    }
    else {
        lampEnabled = true;
        console.log("lamp turns on")
    }
}

// 'a' turns lamp on or off
// 'b' controls strength
client.on('message', (mtopic, message) => {

    const jsonMessage = JSON.parse(message.toString());
    // console.log(`Received message on topic ${mtopic}:`, jsonMessage);

    if (containsValue(jsonMessage, 'a')) {
        controllLamp(lampEnabled)
        // 5 second timer. Gets disabled if lamp gets manually disabled before
time is over
        if (lampEnabled && !timeoutId)
            timeoutId = setTimeout(() => controllLamp(true), 5000);
        else if (!lampEnabled && timeoutId) {
            clearTimeout(timeoutId);
            timeoutId = null;
        }
    }
    else if (containsValue(jsonMessage, 'b') && lampEnabled) {
        strengthLevel++;
        if (strengthLevel > 5)
            strengthLevel = 0;
        console.log('Light strength: ' + strengthLevel)
    }

});

```

Automatisch uitgaan Sensor:

```

const mqtt = require('mqtt');
const readline = require('readline');
const conMessage = 'Connection established'

```

```
const brokerUrl = 'mqtt://broker.hivemq.com';
const topic = 'mqttBewegingIoTHHS';
const client = mqtt.connect(brokerUrl);

let lampStatus = false;
let timeoutId

function containsValue(obj, value) {
  for (const key in obj) {
    if (obj[key] === value) {
      return true;
    }
  }
  return false;
}

function lampDisabler(lampBool)
{
  lampStatus = false;
  client.publish(topic, JSON.stringify({ 'sending command to disable light': 'a'
}), (error) => {
    if (error) {
      console.error('Failed to publish message:', error);
    } else {
      console.log('Message published successfully');
    }
  });
}

client.on('connect', () => {
  console.log('Connected to MQTT broker');

  client.publish(topic, JSON.stringify({ 'Connection Established': conMessage
}), (error) => {
    if (error) {
      console.error('Failed to publish message:', error);
    } else {
      console.log('Message published successfully');
    }
  });

  // Set up readline to listen for key presses
  const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
  });

  rl.on('line', (input) => {
    if (containsValue(input, 'a'))
    {
      lampStatus = !lampStatus;

      // code to handle lamp turning off after 5 seconds
      if (lampStatus && !timeoutId)
```

```

    {
        timeoutId = setTimeout(() => lampDisabler(), 5000); // Delay in
milliseconds
    }
    else if (!lampStatus && timeoutId)
    {
        clearTimeout(timeoutId);
        timeoutId = null;
    }
}
client.publish(topic, JSON.stringify({'Movement found on': input}),
(error) => {
    if (error) {
        console.error('Failed to publish message:', error);
    } else {
        console.log('Message published successfully');
    }
});
});
});
});

```

18. Welke variant lijkt jou het handigst in de praktijk? Geef een uitleg/

- Ik vind de variant op de lamp zelf het handigst vanaf de programmeerkant gezien. De lamp beheerst dan zelf waardoor die aan of uit kan gaan en de sensor hoeft niet bij te houden of de lamp aan of uit staat.

19. In een lokaal zijn 2 lampen en 1 bewegingssensor. Demonstreer dat je met 1 bewegingssensor 2 lampen kan aansturen zonder je code aan te passen uit de vorige vragen

The image shows three terminal windows side-by-side. Each window shows a successful connection to an MQTT broker and a successful message publish. The first window is for 'motionSensor.js', the second for 'lamp.js', and the third for another 'lamp.js' instance. The messages published are 'a' and 'b'.

20. Werk nu samen met een medestudent. Laat jouw lampen aansturen door een bewegingssensor die op de PC van je medestudent draait.

The image shows two terminal windows. The first window shows the execution of 'test.js' which connects to the MQTT broker and publishes messages 'a' and 'b' successfully. The second window shows the execution of 'lamp.js' which connects to the MQTT broker, subscribes successfully, and then outputs 'lamp turns on' and 'lamp turns off'.

21. In een gang staan nu aan beide kanten een bewegingssensor plaatsen die een lamp aanstuurt. Zodra iemand de gang inloopt (maakt niet uit welke kant) gaat de lamp aan (en na x seconde weer uit). Welke variant van opgave 16 werkt nu altijd goed

- de lamp, door de redenen gegeven in opgave 18.