Getting started with Node (Day-2)

Node - Getting Started

1) Complete the *learnyounode*-exercises up until (including) part 6

Exercises that draws on the mosh-examples from "Learn Node in one Hour"

These exercises all assume you have watched the video "Learn Node in one hour" and also my intro-video for today's lecture, in order to understand the "business model" for the following exercises.

1) Simple OS-info file

Create a javascript file that, using nodes CommonJS module system (require/exports), will export an object with the following info (demonstrated for a Window PC)

```
{
  platform: 'win32',
  osType: 'Windows_NT',
  freeMemory: 1244311552,
  totalMemory: 8251834368,
  EOL: '\r\n'
}
```

Create a simple test file that should import (require) the object and print it in a console.log-statement

2) Simple DOS-detector file

Create a file *dosDetector.js* and paste in the code below. It's the start code for an *event-based* control which should fire (emit) an event "DosDetected" if the same URL is added more than once before the time-interval TIME_BETWEEN_CALLS has expired.

```
class DOS_Detector {
   constructor(timeValue){
     super();  //Figure out what it is you have to extend (use moshes video)
     this.urls = new Map();
     this.TIME_BETWEEN_CALLS = timeValue;
   }
   addUrl = (url) =>{
     const time = new Date().getTime();
     if(this.urls.has(url)){
       const deltaTime = time - this.urls.get(url)
       if(deltaTime < this.TIME_BETWEEN_CALLS){
         console.log("TODO: Fire the 'DosDetected' event")
         //Add this info to the event {url:url,timeBetweenCalls:deltaTime}
       }
     }
     this.urls.set(url,time);
   }
}
// Export the class using nodes CommonJS module system (require/exports)
```

Create a simple test file that should import the class, make an instance, and test the behaviour by adding the same URL more than once (use setTimeout to make the second call)

Hints: Observe how this code uses JavaScripts Map (not the map-method on an Array, but the type Map) to store URLs, and how the URL itself is used as the key.

3) Simple WEB/REST-server using functionality from 1+2

Create a new file nodeServer.js and add the following code to the file. Start the server, and verify that you can access the root page via localhost:3000

```
const http = require('http');
const server = http.createServer((req, res) => {
  if (req.url === '/api/os-info') {
    res.setHeader('Content-Type', 'application/json');
    //Return a response with OS-info, using the code implemented in part-a
    return res.end();
  }
  if (req.url === '/') {
    res.setHeader('Content-Type', 'text/html');
    res.write(`<h2>Simple node HTTP server demo</h2>
      <p>Exposes this endpoint <code>/api/os-info</code></p>
    `);
    return res.end();
  }
});
server.on('connection', (sock) => {
  // You can get the client-IP in here, using sock.remoteAddress)
});
server.listen(3000);
console.log('listening on 3000');
//Register for the "DosDetected" event and console.log the url and time info.
```

Add the necessary changes to complete:

- The `/api/os-info` endpoint
- The DOS-detection feature