* Wat betekent foutmelding ‘failed to get input stream from socket’
  + Veel meldingen hiervan met
    - Sender interface mi\_NummersVoorstellen\_OS
    - Receiver interface mi\_NummersVoorstellen\_IS,
    - channel cc\_I2C200\_NummersVoorstellenMP\_SOAP\_Sender
  + Mogelijk staat er niks te luisteren naar berichten.
* “String index out of range: 0”
  + Session ID wordt neit gevuld. Mogelijk is de plek waar sessionID vandaan wordt gehaald net leeg. (onwaarschijnlijk, gezien de frequentie van berichten)
  + Is al vaak aangekaart bij postNL. Niet belangrijk.
* Verwijderen interfaces twee weken geleden: wat ging mis
  + Niks eigenlijk

**Eslint**

* npm i eslint-config-airbnb-base eslint-config-oliver-base eslint-plugin-import
* in project:
  + .eslintrc file
    - { “extends”: “oliver-base” }
* In package.json:
  + “scripts”: { “lint”: “eslint .” }

**Git**

* Create an initial document in a folder.
* Go to that folder in the terminal
  + Initialize the git project: git init. File in working directory
  + Keep track of the status: git status
  + If in the working directory, make a change to the file and save it. Before saving adding it to staging area you can undo the change
    - Git checkout HEAD filename
      * Close and open filename to see that indeed the changes were undone
* Start tracking changes to the file in that folder: add file to staging area
  + Git add filename.extension (eg text.txt; can be more than 1 file)
  + If accidentally made a change in the staging area, reset the file in the staging area to be the same as the HEAD commit (does not discard changes made in working directory)
    - Git reset HEAD filename
* After changing something to the file, show what has changed:
  + Git diff filename
* Permanently store changes from the staging area in the repository
  + Git commit -m “Description of what’s done”
  + Commit you’re currently on is the HEAD commit
    - Git show HEAD
* Refer to a previous version
  + Git log
  + Go to a previous version
    - Git reset code (first 7 chars from a “git log” code)

Connect to github:

* + Uername: reinoutmoes@hotmail.com
  + Password: generate personal acces token in github -> settings -> developer settigns

Branching

* Show on which branch (\* show which one)
  + Git branch
* Create new branch
  + Git branch new\_branch
* Switch to another branch
  + Git checkout other\_branch
* Include all changes on a sub-branch to the master branch
  + Git merge branch\_name
  + Note merge conflicts
* Delete a branch
  + Git branch -d branch\_name

**Heroku**

1. Setup

* Npm install Heroku/brew/Heroku
  + Installs Heroku command line interface (CLI)
* Heroku login
  + To log in on Heroku (shows message in terminal)
  + This authentication is needed for Heroku and git commands to work
  + Check versions (and that they are installed)
    - Node –version / npm –version / git –version

1. Prepare app

* Git clone <https://github.com/heroku/node-js-getting-started.git>
* Cd node-js-getting-started
  + Clones a local version of sample application that you can deploy to Heroku
  + After this: you have a functioning git repository containing simple application and package.json

1. Deploy app to heroku

* Create app on Heroku, which prepares Heroku to receive the source code
  + Heroku create
    - Creates app, but also a git remote (called Heroku) which is associated with local git repositroy
* Deploy code
  + Git push heroku master
* Ensure at least one instance of the app is running
  + Heroku ps:scale web=1
  + Visit app at the url generated by its app name
    - Heroku open

1. View logs

* Heroku logs --tail

1. Define a procfile

* The procfile can be found in root of the project and specifies what command is used to start the app (and what process type, eg web)

1. Scale the app

* Change the number of dynos that are running (default 1)
  + Dyno is a container that runs the command specified in the Procfile
  + Up/down scaling:
    - Heroku ps:scale web=# with # number of dynos (0, 1, or more)

1. Declare app dependencies

* Heroku recognizes an app by the existence of a package.json file in the root directory. Create one with
  + Npm init --yes
  + Package.json determines version of node.js and the dependencies that should be installed with your application.
  + When app is deployed, Heroku reads package.json file and installs appropriate node version and dependencies with npm install

1. Run app locally
   1. Heroku local web
      1. Command installed as part of Heroku CLI
      2. It examines Procfile to determine what to run
      3. Open localhost:XXXX to see the app running locally
      4. In CLI, press Ctrl + C to exit to stop the app running
2. Push local changes

* Change something in the file
  + Npm install (if adding new packages)
  + Heroku local -> refresh page
* Deploy to Heroku
  + Git add .
  + Git commit -m “Description”
  + Git push heroku master

1. Provision add ons = third party cloud services that provide out-of-the-box additional services for application.

**Sourcetree**

1. Maak een repository in github.
   1. Verschijnt als remote repository in de lijst in sourcetree (‘show repository browser’ onder ‘window’ in sourcetree)
2. Clone de repository
   1. Wachtwoord is macbook toegangs ww
   2. Creeert map op gespecificeerde locatie
3. Sleep bestanden naar die map
4. Open de repository in sourcetree (klik in de local repository lijst op de juiste repository)
   1. Committen en pushen

**Sourcetree and heroku**

1. Add heroku git remote. Deploy new versions of the app by pushing code changes to a Heroku-hosted git repo. For this to work, you must have *local* Git repo to know the URL of the Heroku-hosted repo. Add the Heroku0hosted repo as a Git remote with
   1. Heroku create
      1. Heroku remote -v for seeing the version and name
      2. Heroku remote rename oldname newname for renaming
2. Add a procfile. Commit text file to app’s root directory named Procfile (WITHOUT file extension). Tells Heroku which commands to run to start the app.
   1. For webscraper I use web: npm run dev (plain text in Procfile).
   2. Ordinarily however, it is web: node app.js
3. Listen on correct port. On Heroku, the app must listen on a specific port.
   1. I had this already right:
      1. let port = process.env.PORT || 3000

app.listen(port)

1. Use database or object storage instead of writing to local filesystem
2. Deploy (Node.js specific)
   1. Declare app dependencies in package.json, with npm init
   2. Specify version of node. Should be defined in package.json file.
      1. node --version to find the version
      2. Then check if package.json matches that (“engines”: …)
3. Specify a start script (Procfile)
4. Build app and run it locally.
   1. npm install to install the dependencies declared I package.json
      1. Think this is obsolete
   2. heroku local web to start app locally
5. Build .gitignore file
   1. /node\_modules
   2. Npm-debug.log
   3. .DS\_Store
   4. /\*.env
6. Deploy application to Heroku
   1. Commit changes to git
      1. Git add .
      2. Git commit -m “description”
   2. Heroku login
   3. Heroku create
   4. Git push Heroku master
   5. Heroku open to open app in browser