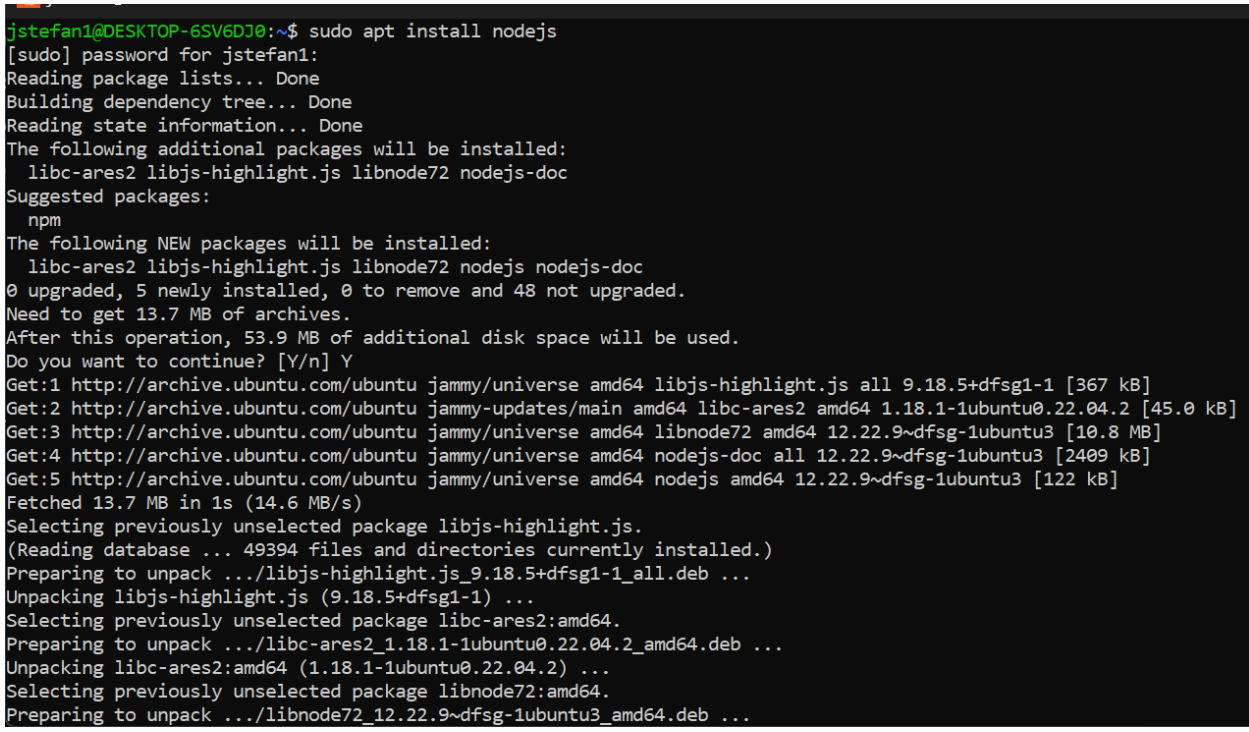
# CPE 322 – Lab 6

1. First, I installed Node.js:
   a.


2. I then ran the following commands and files in the linux terminal (Ubuntu):

a.

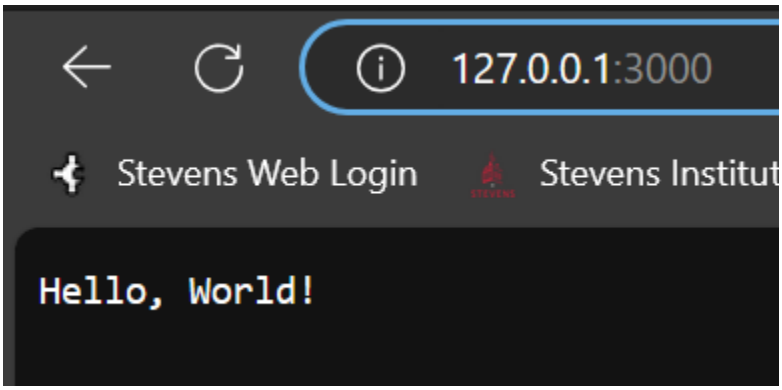
```
jstefan1@DESKTOP-6SV6DJ0:~$ node -v
v12.22.9
jstefan1@DESKTOP-6SV6DJ0:~$ npm -v
10.1.0
jstefan1@DESKTOP-6SV6DJ0:~$ node -h
Usage: node [options] [ script.js ] [arguments]
       node inspect [options] [ script.js | host:port ] [arguments]

Options:
  -                                  script read from stdin (default if no file name is
                                     provided, interactive mode if a tty)
  --                                 indicate the end of node options
  --abort-on-uncaught-exception      aborting instead of exiting causes a core file to
                                     be generated for analysis
  -c, --check                        syntax check script without executing
  --completion-bash                  print source-able bash completion script
  --conditions=...                   additional user conditions for conditional exports
                                     and imports
  --cpu-prof                         Start the V8 CPU profiler on start up, and write
                                     the CPU profile to disk before exit. If
                                     --cpu-prof-dir is not specified, write the profile
                                     to the current working directory.
  --cpu-prof-dir=...                 Directory where the V8 profiles generated by
                                     --cpu-prof will be placed. Does not affect --prof.
  --cpu-prof-interval=...            specified sampling interval in microseconds for the
                                     V8 CPU profile generated with --cpu-prof. (default:
                                     1000)
  --cpu-prof-name=...                specified file name of the V8 CPU profile generated
                                     with --cpu-prof
  --diagnostic-dir=...               set dir for all output files (default: current
```

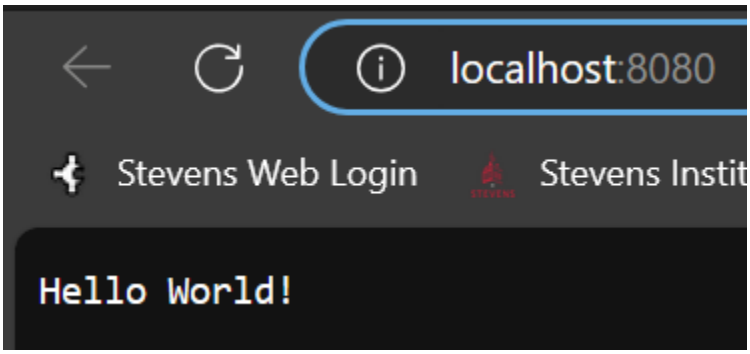
b.

```
jstefan1@DESKTOP-6SV6DJ0:~/iot/lesson6$ node hello-world.js
Server running at http://127.0.0.1:3000/
```

c.

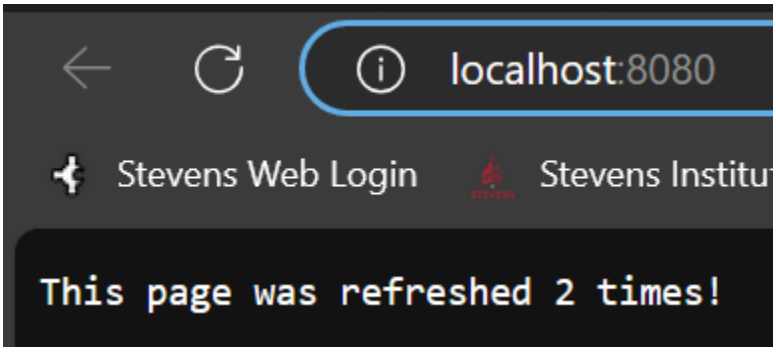


d.
```
jstefan1@DESKTOP-6SV6DJ0:~/iot/lesson6$ node hello.js
Server running at http://127.0.0.1:8080/
response end call done
request end event fired
response end call done
request end event fired
```

e. 


f.
```
jstefan1@DESKTOP-6SV6DJ0:~/iot/lesson6$ node http.js
0

1
```

g. 


3. Installing Pystache:

a.
```
jstefan1@DESKTOP-6SV6DJ0:~$ sudo pip3 install pystache
[sudo] password for jstefan1:
Collecting pystache
  Downloading pystache-0.6.5-py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.3/81.3 KB 2.8 MB/s eta 0:00:00
Installing collected packages: pystache
Successfully installed pystache-0.6.5
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packa
ge manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

4. Then I ran the following commands and files:

a.

```
jstefan1@DESKTOP-6SV6DJ0:~$ cd ~/iot/lesson6
jstefan1@DESKTOP-6SV6DJ0:~/iot/lesson6$ cat say_hello.mustache
Hello, {{to}}!
```

b.

```
jstefan1@DESKTOP-6SV6DJ0:~/iot/lesson6$ cat say_hello.py
# https://github.com/defunkt/pystache
import pystache
print(pystache.render('Hi {{person}}!', {'person': 'Alexa'}))

# Create dedicated view classes to hold view logic
class SayHello(object):
    def to(self):
        return "World"
hello = SayHello()

# Use template in say_hello.mustache
renderer = pystache.Renderer()
print(renderer.render(hello))

# Pre-parse a template
parsed = pystache.parse('Hey {{#who}}{{.}}!{{/who}}')
print(parsed)
print(renderer.render(parsed, {'who': 'Google'}))
print(renderer.render(parsed, {'who': 'Siri'}))
```

c.

```
jstefan1@DESKTOP-6SV6DJ0:~/iot/lesson6$ python3 say_hello.py
Hi Alexa!
Hello, World!

['Hey ', _SectionNode(key='who', index_begin=12, index_end=18, parsed=[_EscapeNode(key='.'), '!'])]
Hey Google!
Hey Siri!
```