Installing Node.js

begin by creating a directory in /opt for node to go

```
sudo mkdir /opt/node
```

Next get the binary package

```
wget http://nodejs.org/dist/v0.10.2/node-v0.10.2-linux-arm-pi.tar.gz
```

then unpack the tar compression

```
tar xvzf node-v0.10.2-linux-arm-pi.tar.gz
```

finally copy the packages into the new /opt/node directory

```
sudo cp -r node-v0.10.2-linux-arm-pi/* /opt/node
```

Node can be started with the following command

```
sudo /etc/init.d/nodejs.sh start
```

the version of node can be determined with the command

```
node -v
```

Creating directory for scripts and running scripts

```
mkdir /Desktop/node
```

I tested a simple server example named app.js and added it to this new directory.

execute by having the file name as a parameter in the node program.

```
node app.js
```

Overcoming obstacles with Mongodb

Mongodb's support for non x86 architectures including the ARM processor is fairly limited and requires some manual installation in comparison to on standard computers. The following command will download the mongo packages for the raspberry pi.

git clone https://github.com/skrabban/mongo-nonx86

One notable difference is the time it takes for several precesses to occur after download is a matter of hours. The commands that trigger these long processes are:

```
sudo scons
```

```
sudo scons --prefix=/opt/mongo install
```

after they were completed several init files had to be separately downloaded and put into specific directories. I ran into serious issues at this stage that I was unable to resolve.

I did not want to give up on Mongo and switch to a relational database but I was not going to get it to run on the raspberry pi. I decided to install Mongo on a Ubuntu system I have running at my home which is a standard x86 machine.

The first thing that must be done in the mongo installation is add a key for the packages that will be downloaded using APT.

```
    sudo apt-key adv --keyserver hkp://
keyserver.ubuntu.com:80 --recv 7F0CEB10
```

and add a source list file

```
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
```

after these modifacations update APT.

```
sudo apt-get update
```

after this mongo is ready to be installed

```
sudo apt-get install mongodb-10gen
```

now mongo can be started my simply typing “mongo” into the command line.

Tutorial used: http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/

Accessing mongodb remotely from node.js

Instead of accessing a database on the same machine as the script which is referred to as localhost we will be connecting to the Ubuntu machine for the database from the node framework on the Raspberry Pi.

The only additional thing that needs to be downloaded is the mongo module that should be in the same directory of the node scripts implementing it. once this is done the scripts can be run. Within the script on github called app.js accesses the remote database with the specified ip address and port and input and query from the database.

Response sent back to client

The beginning of the response comes from a static html file named intro.html It contains the html and css required to make the the table responsive to the screen size. Then the rest of the response, which is just the actual table with all the tags is dynamically generated after the database has been queried. The response is sent once the response.end() function is called.

Responsive Table

The table switches between a 2 column table with fields, Hour and Temperature to a table who's columns break down with labels on the side. This is all done in css. When switching, the css hides the header of the table containing the field names and uses

another line in he css to know which titles to add to the side of each row. This happens automatically when the page is resized.