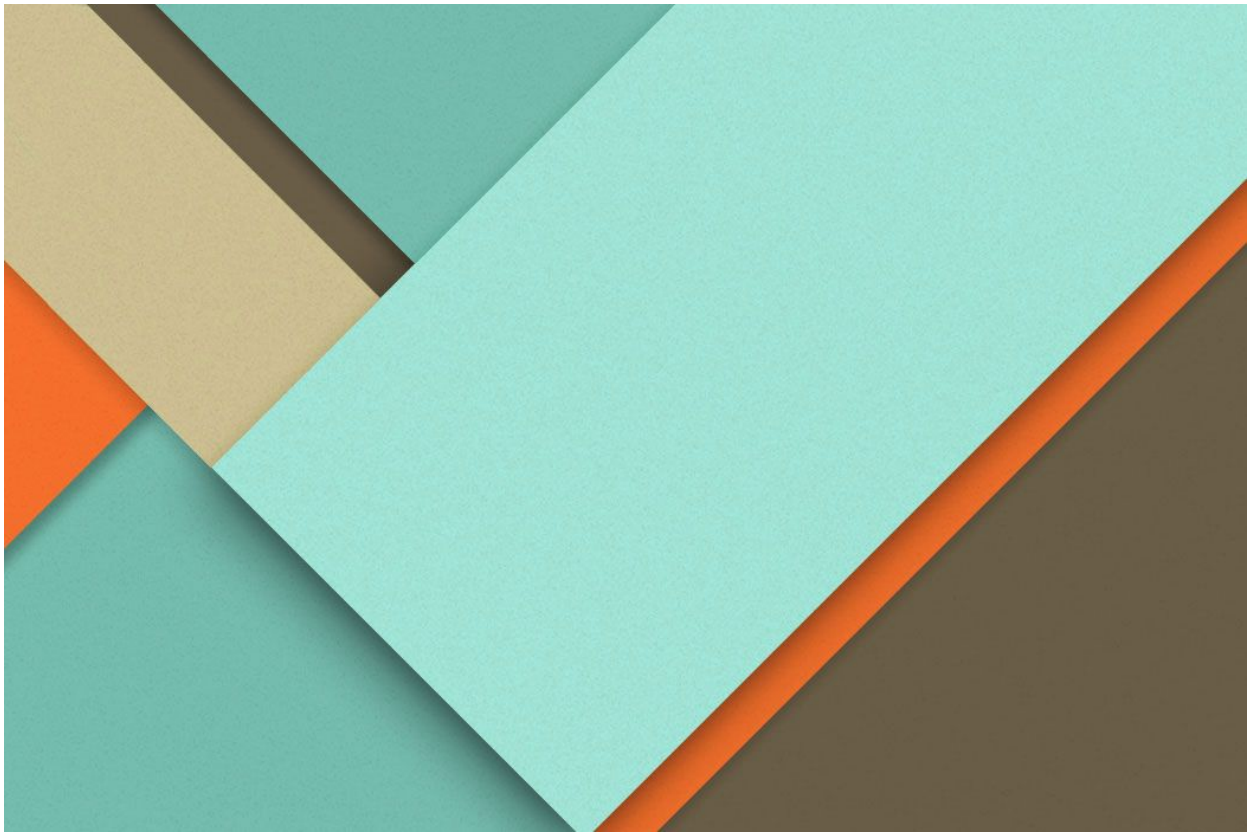

Learning Notes

Bridging NodeJS & Python

Using Unix or Windows Socket with IPC (Inter-Process Communication)

Haris Hashim

harishashim@gmail.com

Table of Contents

Lesson 1: Setting Up JS Development Environment

A. Visual Studio Code

1. Download and install Visual Studio Code.
2. Download and install Node.
3. Some tutorial to go through before the learning session (tutorial 1 first and then 2).
    - Tutorial 1: Node.js Tutorial in VS Code
    - Tutorial 2: Debugging Node.JS with VS Code
4. Exploring VSCode
    - Find a folder (or create new one) to use as project folder or what is called as project workspace. Right click to open it using VSCode.
    - Open VSCode terminal (CTRL-` or Control BackTick).
    - Run “npm init” in terminal to create node js project file, which is the package.json file.
    - learning about package.json and dependency added into package.json.
    - To add dependency run “npm install <package> -save”, this will download dependency and add it to package.json file.

B. Quick Hands-On with Node.JS

1. Looking for library and sample code online
    - Google for “node ipc” to find NPM for node-ipc.
    - Read documentation.
    - Visit github page for node-ipc and explore the codes.
2. Importing sample code in VSCode
    - Look for the easiest sample code in node-ipc github page and we will import this code into VSCode.
    - The sample code is UnixWindowsSocket/basic with 2 files world-server.js and hello-client.js.
    - Download the code using git clone or download zip and put the 2 files in project folder.
    - Run to see missing dependency.
        - The command for server was “node world-server.js” to run IPC server.
        - The command for client was “node hello-client.js” to run IPC server.
    - The missing dependency was node-ipc. To install it run “npm install node-ipc -save”.
    - There was another problem, the import statement (or in JS, require) path was wrong. Correct the code to

```
const ipc=require('./node_modules/node-ipc');
```

    - Correct the same code in hello-client.js.
    - Run both server and client code to see IPC server and client talking to each other.

Lesson 2: IPC Communication in Node JS

A. Using node-ipc library and understanding the code.

1. Refer to example code.
    - Original server Code in Github

```
ipc.config.id = 'world';
ipc.config.retry= 1500;

ipc.serve(
    function(){
        ipc.server.on(
            'app.message',
            function(data,socket){
                ipc.server.emit(
                    socket,
                    'app.message',
                    {
                        id      : ipc.config.id,
                        message : data.message+' world!'
                    }
                );
            }
        );
    }
);



ipc.server.start();
```

- Server code after we break it down into function and var for better understanding

```
ipc.config.id = "world";
ipc.config.retry = 1500;

function theCallback() {
  console.log("This is the callback");

  ipc.server.on("app.message", theEventCallback);
}

function theEventCallback( data, socket) {
    var jsonData = {
        id: ipc.config any
        message: data.message + " world!"
      };

  ipc.server.emit(socket, "app.message", jsonData);
  console.log("This is the event callback!");
}

ipc.serve(theCallback);

ipc.server.start();
```

2. Review the following functionality for both server and client code. Refer NPM or Github documentation for more information:

- ipc.serve
- Ipc.server.on <custom> event
- ipc.server.emit
- ipc.connectTo
- ipc.of.<socket_id>.on 'connect' event
- ipc.of.<socket_id>.on 'disconnect' event
- ipc.of.<socket_id>.on <custom> event

B. Take Home Exercise

Self study how to implement IPC communication using python and communication between python client with Node.js server.

Lesson 3: Consuming HTTP REST Service

A. Designing message passing protocol for IPC

B. Using HTTP component in Node.JS

C. Understanding REST API design

D. Consuming REST API

E. Exercise

Finish the implementation of python application communicating with Node.JS using IPC which then consume REST API from the cloud