Introduction

Download sample scripts:

```
git clone https://github.com/raymondwcs/http-samples.git
```

Part (1) Components of HTTP Request URL

Identify the pathname & query string in below URL.

```
http://apilayer.net/api/live
    ?access_key=e0d485961e1cb71d4e228cef340d764c
    &currencies=EUR,GBP,CAD,PLN
    &source=USD
    &format=1
```

Part (2) HTTP Response Headers

1. Study `serveJPGFile.js`
2. Modify `serveJPGFile.js` to accept HTTP GET requests for downloading a PDF file (in addition to downloading a JPG file)
3. What could happen if you don't set the `Content-Type` field to `application/pdf` in the response header?

Part (3) HTTP POST requests

1. Study `greetings2.js`
2. Prepare a HTML form similar to the one shown in below:

   Your name:

   ☐ Show current time

   Submit

3. Modify `greetings2.js` to accept the above HTML form via HTTP **POST** request
4. Use `curl` to test/verify your modified script.

   Usage of `curl` (POST):

   ```
   curl –v –X POST http://127.0.0.1:8099 -d "<query string>"
   ```

Part (4) – *Design* and implement a Node.JS server that offers simple interest calculation service

Requirements

Your Node.JS should allow clients to make **HTTP GET** requests to:

1. Calculate simple interest (I) using the simple interest formula[1]: **I = Prt** where P is the principal amount of money to be invested at an interest rate r% per period for t number of time periods.
2. Client requests must include principal (P), interest rate (r) and the number of time periods (t) in the query string.
3. Allow client to choose result format:
   - JSON
     `{principal: xxx, rate: yyy, time: zzz, interest: kkk}`
   - HTML

   Default format is JSON
4. Deploy your app to IBM Bluemix (or Heroku)
   - Prepare a `package.json` file for your server app

Design Tips

1. Determine the **path name** for HTTP GET requests
2. Determine the **query string**

Implementation Tips

1. Use `greetings1.js` as template. Name your app as `server.js`
2. Create a HTML form that would allow the user to enter P, r and t and see the result in HTML (or in JSON)
3. Use `curl` (in addition to Web browser) to help you verify your implementation:

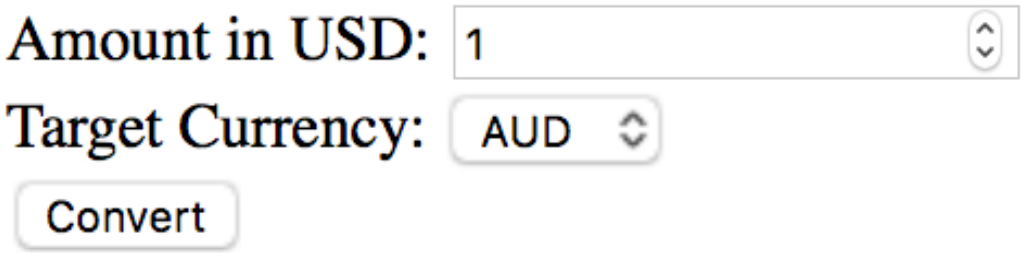
```
curl –v –X GET "http://127.0.0.1:8099/<path-name>?<query-string>"
```

[1] http://www.wikihow.com/Calculate-Simple-Interest

Part (5) – Server-side HTML generation

Recall the currencyXchg.js that you have written in the previous lab exercise. Modify it to address the below requirements.

1. Upon receiving `GET /`, your server returns the below HTML form



   The HTML form allows clients to specify an amount in USD to be converted to either AUD or CAD.

2. Your server returns `404` for all other requests.

Here's a sample of the modified `currencyXchg.js`:

`currencyXchg.mybluemix.net`