Approach Lecture –

Node Pages

Objectives: exploring server fundamentals

* Use XMLHTTPRequest API to transfer data
* Setting up HTTP Transaction
* Connect the client and server to make sure requests are correctly handled

1. XMLHTTPRequest?
   1. Web API (deprecated)
   2. Used to interact with serers, built into the browser
   3. This is an **object** with various methods that llow for transfer of data between web browser and server.
   4. Can be used with protocols other than HTTP; data can be not only XML, but also JSON, HTML, or plaintext.
   5. XMLHTTPRequest properties:
      1. Onreadystatechange
      2. readyState
      3. responseText
      4. responseXML
      5. status
      6. statusText
   6. XMLHTTPRequest methods
      1. Abort()
      2. getAllResponseHEaders()
      3. getResponseHEader()
      4. open()
         1. specifies the type of request, URL, and other optional attributes..
      5. send(string)
2. Node modules
   1. http
      1. allows node to transfer data over http
   2. fs
      1. allows you to work with the file sys on your computer
   3. path
      1. provides utils for working with files.
3. Fs methods
   1. readFileSync
      1. locates and reads a given file, sends it back to the client
      2. There isn’t a particular thing we have to do asynchronously, so we use the synchronous option.
   2. appendFileSync
      1. synchronously adds given data to a file. If file doesn’t exist, it creates it.
   3. Fs.writeFileSync
      1. Overwrites data in a specified file.
   4. Fs.unlinkSync
      1. Deletes file from your directory.
4. Path methods
   1. Path.join – provides a simple concatenation
   2. Path.resolve – creates an absolute path from the root
      1. It looks for the first segment with/from the right and append everything up to this point to the root.
   3. \_\_dirname – provides the absolute directory to the current file.
5. HTTP Transactions
   1. How do HTTP Requests work?
6. http.createServer()
   1. creates our server
   2. We turn our machine into an HTTP server by creating an HTTP server object.
   3. This object is what .listens to ports on the computer and executes a function (request handler) when an HTTP request is received.
7. The Request Object
   1. Method
      1. GET/POST/PUT/DELETE
   2. URL
      1. Address where a resource is located
   3. Headers
      1. Metadata to provide info about the request.
   4. Payload
   5. In the case of a POST/PUT request,
      1. We can use the request body to transfer the data from the client to the server in a readable stream.
      2. We grab the data out of this stream by listening (using .on) to the stream’s data and end events.
   6. The buffer stream can be displayed as a string, but its type will be different and not deeply equal to a particular string unless the buffer is passed through a toString method.

*Common syntax for writing error Code and sending an error message to the user.*

*Within a try/catch block:*

*return* response.writeHead(errorCode, responseHead).end('Error Message')