Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your report for each of the technologies you use in your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we'd like to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.
- **Who worked with this?**: It's not necessary for the entire team to work with every technology used, but we'd like to know who worked with what.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

HTTP

General Information & Licensing

Code Repository	https://github.com/ethan-richardson/Bullboard/tree/jay
License Type	MIT
License Description	Copyright Joyent, Inc. and other Node contributors.
License Restrictions	Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included

	in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Who worked with this?	Jay Hou

Use as many of the sections below as needed, or create more, to explain every function, method, class, or object type you used from this library/framework.

http.createServer(function(request, response)

Purpose

- Creates HTTP server for the web application and returns a new instance of http.Server. It creates and instantiates the HTTP server to receive requests and also return responses.
- Used in line 7 of server.js

Magic

http.createServer is used to create a http server that takes in messages and also can return responses. It returns a new instance of http.Server which is an extension of net.Server, which is used to create a TCP or IPC server that also extends EventEmitter. EventEmitter is a class that is defined and exposed by event's modules. This line is used in line 7 of server.js in the frontend/pages/ directory. Else we would need to create a net server in order to send responses and get requests.

res(ponse).setHeader

Purpose

- This line of code helps set the headers for the HTTP response.
- Used in line 8 of server.js

Magic

res.setHeader is used to set the “X-Content-Type-Options” to “nosniff.” This function returns the response object and sets the single header value for implicit headers. We can use this to set content-type or any of the html headers. If we did response.setHeader('Content-Type', 'text/html'); It would set Content-Type to “text/html” or else we would have to manually type these lines out bit by bit and send it over as a response. We would have to add them manually by adding “X-Content-Type-Options: “ + “nosniff” to a response and sending that response by itself.

res(ponse).writeHead

Purpose

- This line of code is used to help define mimetype of the response and can also be used to set other headers in the html response.
- It is used in line 12, 19, 30, 37, 48, 56, 66 of server.js

Magic

res.writeHead is used to set the mimetype amd response code for the responses and also used to set other headers that the user may want that response to have set. In the case of how I used it, I always used it to set the content-length and content-type of the html response. This sends a response header to the request and the status code is the 3-digit HTTP status code. It returns a reference to the ServerResponse that can also be chained. It must be called before response.end(). Without this function, the response code and content-length and content-type would have to manually be set within the response.
I.e. let response = “HTTP/1.1 “ + “200 OK ” + “\r\n”;
response += “Content-Type: “ + mimeType + “\r\n”;
response += “Content-Length: “ + length of content + “\r\n\r\n”;

res(ponse).end()

Purpose

- This line of code is used to help push data to the response and is what sends the information used to display the content.
- It is used in line 16, 23, 34, 41, 52, 60, 70 of server.js

Magic

res.end() must always be called on each response in order to end the response. However there are parameters that may also be called with response.end([data[,encoding]][,callback]) and when data is specified, it is similar to calling response.write(data, encoding) and then ending the response afterwards. It will send a chunk of the response body and is raw HTTP body and has nothing to do with the higher-level multi-part body encodings. The data is streamed and the data is sent separately. Since we call it with the content, the content will be sent as if it was called as response.write(data) and the response will end shortly after that data is sent. If we were unable to send the content with res.end(data), we would have to use response.write or manually parse the data and send it through the response.