Battelship Game Server Sent Events
S19 CSCI470 Web Science

Assigned: 2019-02-25
Reference: JavaScript, CSS, HTML5, Ajax, JSON Due Date: 2019-03-08

Summary:

Abstract:	Battleship Gameboard Server Sent Events
Objectives:	1. HTML5 API SSE 2. EventSource object 3. Event handling 4. JavaScript for Behaviors 5. JSON and AJAX 6. Node
Grading:	45 pts - A $\geq$ 41.85; A- $\geq$ 40.50; B+ $\geq$ 39.15; B $\geq$ 37.35; B- $\geq$ 36; C+ $\geq$ 34.65; C $\geq$ 32.85; C- $\geq$ 31.75; D+ $\geq$ 30.15; D $\geq$ 28.35; D- $\geq$ 27
Outcomes:	R2 (CAC-a,c,i, j, k; EAC-a, b, c, e, k, 1, 2); R5 (CAC-a, i; EAC-a, k) (see syllabus for description of course outcomes)

Project Description:

Create a Server Sent Events set of end-points in the Node server to provide updates to the player's board game as the two Node servers do battle. The server sent events provide a `data:` message which can be captured using the `message` event on the `EventSource` object in the browser. Additionally, you may elect to trigger events from the server using the server sent events by also sending a `event:` as part of the message to the browser. The text following the event can then be used as an event through the `EventSource` object in the browser.

Obtaining Project Files:

1. Logon to `gitlab.cs.mtech.edu` and locate the project `bsSSE` under the |S19 CSCI470| (sub)group, and then fork this project into your own account.
2. Navigate to your own account and locate the `bsSSE` project to just forked, and copy the project url
3. Next, logon to `csdept16.mtech.edu` using your Department username and password.
4. Execute the `mkdir ~/CSCI470/Projects/` command, which will create a projects folder if not already created.
5. Execute the `cd ~/CSCI470/Projects` command, which will change the current working directory to the specified parameter.
6. Issue the command `git clone <project_url>`, where `<project_url>` is the url you copied in the above step. This will create the project folder inside your `~/CSCI470/Projects` directory,
7. Execute the command `cd bsSSE` to enter the project directory.
8. Continue with the specific project activities below.

Project Activities:

Please perform the following activities in the completion of the lab assignment.

1. Start your server by executing the command `npm start` from the `bsServer` directory.
2. Navigate your browser to

   `http://csdept16.mtech.edu:30120`

   where `30120` is your port id, and your should see a welcome message from express.
3. Familiarize yourself with the contents of the `app.js` and the `sse.js` under the `routes` directory.
4. The provided code will stream random tiles from a valid board with a default layout of the ships. This can be used to test your game board and its redraw functionality once you consume the messages from your browser.
5. Modify your game board to:
   (a) Create an `EventSource` object and connect to the end-point `/sse/stream`.
   (b) Add an event listener for at least the `message` event on the `EventSource`. You should also consider adding other events - one for each ship - to the `EventSource` object.
   (c) When each event is received, you should process the event; registering either a his or a miss on the corresponding tile. This should result in the game model being updated, triggering either automatically or through additional explicit coding, a redraw of the game board to reflect the new information.
6. Modify the node `sse.js` routes:
   (a) Feel free to modify the time interval between invocations of the `sendSSEMSG()` function to see how well your game board logic works at updating the information.
   (b) Move the invocation behavior our of the `/stream` route such that the `/stream` route only initiates the sse session with the client, but messages sent to the client are performed in a different function.
   (c) Pay attention to the following:
      - Keeping track of the correct `response object` to send SSE messages
      - Ability to invoke the function that sends the sse data to the client from outside of the `sse.js` module.
   (d) Use the new function to replicate the random sending of tiles guesses to the client with the disposition of hit (ship name) or miss.

Figure 1: Programming Project Grading Rubric

Attribute (pts)	Exceptional (1)	Acceptable (0.8)	Amateur (0.7)	Unsatisfactory (0.6)
Specification (10)	The program works and meets all of the specifications.	The program works and produces correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results, but does not display them correctly.	The program produces incorrect results.
Readability (10)	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
Reusability (10)	The code could be reused as a whole or each routine could be reused.	Most of the code could be reused in other programs.	Some parts of the code could be reused in other programs.	The code is not organized for reusability.
Documentation (10)	The documentation is well written and clearly explains what the code is accomplishing and how.	The documentation consists of embedded comments and some simple header documentation that is somewhat useful in understanding the code.	The documentation is simply comments embedded in the code with some simple header comments separating routines.	The documentation is simply comments embedded in the code and does not help the reader understand the code.
Efficiency (5)	The code is extremely efficient without sacrificing readability and understanding.	The code is fairly efficient without sacrificing readability and understanding.	The code is brute force and unnecessarily long.	The code is huge and appears to be patched together.
Delivery (total)	The program was delivered on-time.	The program was delivered within a week of the due date.	The program was delivered within 2-weeks of the due date.	The code was more than 2-weeks overdue.

The *delivery* attribute weights will be applied to the total score from the other attributes. That is, if a project scored 36 points total for the sum of *specification*, *readability*, *reusability*, *documentation* and *efficiency* attributes, but was turned in within 2-weeks of the due date, the project score would be $36 \cdot 0.7 = 25.2$.

Project Grading:

The project must compile without errors (ideally without warnings) and should not fault upon execution. All errors should be caught if thrown and handled in a rational manner. Grading will follow the *project grading rubric* shown in figure 1.

Collaboration Opportunities:

You may collaborate with up to one other person on this project - but you must cite, in the code (html, css, js) each students' contribution.