Assignment 1
CS-455
Spring 2019

Due Date: 3/1/2019 at 11:55 pm (No extensions)

You may work in groups of 4

Goals:

1. **Broader goal:** examine concepts of web application technologies and security issues associated with them.
2. **Specific goals:** to provide introduction to:
   (a) HTTP protocol
   (b) Web application architecture: front-end vs back-end.
   (c) Encoding and transmission of web data using `GET` and `POST` methods
   (d) Web application development using Node.js
   (e) To practice all of the above by implementing a rock-paper-scissors game using Node.js

Overview

You shall be implementing a simple Rock Paper Scissors game using Node.js. Doing so will better familiarize with handling HTTP requests and responses, passing of form data between the front-end and the back-end using HTTP `GET` and `POST` methods, and learn to develop web apps with Node.js.

Web App Specifications

Your web app architecture shall comprise a front-end and back-end.

Front-end: When the user navigates to the homepage, the homepage will display a form consisting of three radio buttons:

- Rock
- Paper
- Scissors

and a `Play!` push button. Figure 1 shows an example. The player shall make a choice and click `Play!`, which will use the `POST` method to send the choice to the Node.js server.

The Back-end: The back-end shall consist of the Node.js server listening on port `3000`. The server shall do the following:

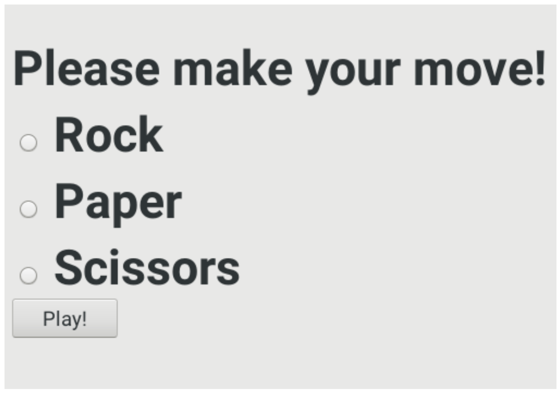


Figure 1: The homepage

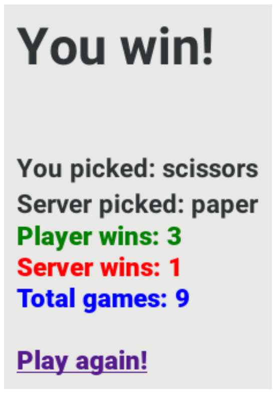
1. Receive the HTTP request
2. Extract the player's choice from the body of the message
3. Randomly choose rock, paper, or scissors
4. Compare the server's choice to the player's choice and apply the rules of Rock Paper Scissors to determine if the player wins, the server wins, or whether the game is a draw.
5. Dynamically generate a page consisting of:
   - The result of the game
   - The server's choice
   - The player's choice
   - The number of player wins
   - The number of server wins
   - The total number of games played since the server was started
   - A link back to the home page allowing the player to play another game.

The Figure 2 gives an example of how the page should look.

The server should be invoked using `node rps.js`. The homepage should be accessible from the front-end by typing `http://localhost:3000` into the browser URL bar.

Resources

- Rules of Rock Paper Scissors https://bit.ly/2DycjNI
- HTML forms and radio buttons https://bit.ly/2gmUh9x
- Generating a random number in JavaScript https://bit.ly/2LWju4M


You win!

You picked: scissors
Server picked: paper
Player wins: 3
Server wins: 1
Total games: 9

Play again!

Figure 2: The game outcome

- To see examples of handling `POST` requests and other Node.js basics please see the sample codes from the Node.js lecture https://bit.ly/2GCpUa2

BONUS:

Extend your program by (1) implementing a game of Rock Paper Scissors Lizard Spock (the rules can be found here https://bit.ly/1ChTjfl); and (2) Use a templating engine to generate the page for displaying the game outcome and statistics (a short tutorial on templating engines can be found here: https://bit.ly/1QRPOV2.

SUBMISSION GUIDELINES:

- This assignment must be completed using Node.js. **You may use `http` or `express` packages. However, no other packages should be used**.
- Please hand in your source code for the html page implementing the front-end and your Node.js JavaScript code. To TITANIUM.
- **If you worked in a group, only one person within each group should submit.**
- Write a README file (text file, do not submit a .doc file) which contains
  - Names and email addresses of all group members
  - How to execute your program
  - Whether you implemented the extra credit
  - Anything special about your submission that we should take note of
- Place all your files under one directory with a unique name (such as `p1-[userid]` for assignment 1, e.g. `p1-mgofman1`).
- Tar the contents of this directory using the following command. `tar cvf [directory_name].tar [directory_name]` E.g. `tar -cvf p1-mgofman1.tar p1-mgofman1/`

- Use TITANIUM to upload the tared file you created above.

Grading guideline:

- Node.js code starts without errors: 5'
- Correct front-end: 15'
- Correct back-end: 75'
- README file included: 5'
- BONUS: 5 points
- Late submissions shall be penalized 10%. No assignments shall be accepted after 24 hours.

Academic Honesty:

All forms of cheating shall be treated with utmost seriousness. You may discuss the problems with other students, however, you must write your **OWN codes and solutions**. Discussing solutions to the problem is **NOT** acceptable (unless specified otherwise). Copying an assignment from another student or allowing another student to copy your work **may lead to an automatic F for this course**. Moss shall be used to detect plagiarism in programming assignments. If you have any questions about whether an act of collaboration may be treated as academic dishonesty, please consult the instructor before you collaborate. Details posted at http://www.fullerton.edu/senate/documents/PDF/300/UPS300-021.pdf.