Assignment 3. Chorus lapilli

Useful pointers

- React – A JavaScript library for building user interfaces
- Hello World for React
- Tutorial: Intro to React
- React source code

Background and motivation

This assignment is designed to let you build an app of your own, so that you can see all its pieces. This is in contrast with your main class project, where your project team will build a client-server application and most likely you will become expert in your part of the app with only nodding familiarity with the rest.

This assignment is supposed to be quite simple, because you should be spending most of your time working on the project. To help keep it simple, you'll do a tutorial on basic React, the JavaScript runtime that is the basis for most of the student projects.

A simple interactive game app

The goal of this assignment is to build a simple application using React.

A secondary goal of this assignment is for you to record what you did. This is so that someone else can build your app, and so that you can see later what you did, to fix your app or to build a similar app.

Start by reading the Hello World guide cited above.

Next, build the simple tic-tac-toe game of the tutorial cited above. Use the tutorial's setup option 2 with local development environment; call your app "`tic-tac-toe`". As you go, keep a log in the file `tic-tac-toe.txt` of what you have done so that you can reproduce the results later. This should not merely be a transcript of what you typed: it should be more like a true lab notebook, in which you briefly note down what you did and what happened. Write down every significant action that you take, including installing and configuring components as well as any code that you write.

Next, take a breather and reread the source code of your app. Although you needn't understand every little detail of what it does, it may well give you pointers about useful things to know about React, JavaScript, JSon, Node.js, HTML, CSS, DOM, JSX, Chrome dev tools, Firefox dev tools React keys, React reconciliation, etc.

Now, use your experience to build an app that lets you play a variant of terni lapilli ("three grains"), a popular board game in ancient Rome. We'll call this variant *chorus lapilli* ("dancing grains") and you should call your app "`chorus-lapilli`".

Chorus lapilli is like tic-tac-toe in that players take turn placing pieces on a 3×3 board and the goal is to get three pieces in a row. However, it differs from tic-tac-toe in two ways:

- After your first three moves, instead of adding further pieces you must instead move one of your existing pieces to an empty square that is adjacent vertically, horizontally, or diagonally. Therefore, after your third move you always occupy three squares.

- If you have three pieces on the board and one of your pieces is in the center square, you must either win or vacate the center square in your next move.

Keep a log of how you built your chorus lapilli app in a file `chorus-lapilli.txt`. Like your other log file, it should be a complete set of steps to build your toy application, that you could give to someone else to reproduce your game.

When you have `chorus-lapilli` working, create a gzipped tarball `chorus-lapilli.tgz` of it by using `npm pack`. Use `npm pack`'s `--dry-run` option before actually running it, and record its output into your log.

Submit

Submit the following files.

- The files `tic-tac-toe.txt` and `chorus-lapilli.txt`.
- The `chorus-lapilli.tgz` tarball.

The `.txt` files should be ASCII text files, with no carriage returns, and with no more than 80 columns per line except possibly for the dry-run output. The shell command:

```
expand tic-tac-toe.txt chorus-lapilli.txt |
    awk '/\r/ || 80 < length'
```

should output at most some dry-run lines.

$Id: assign3.html,v 1.40 2021/01/27 02:55:08 eggert Exp $