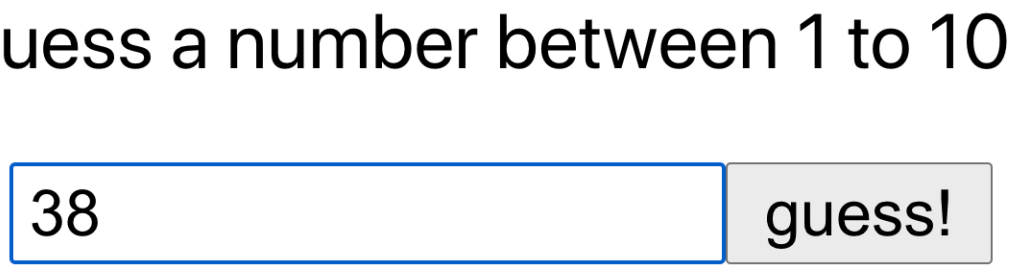

Practice Homework #6
Backend 練習 — Number Guessing Game

Due： 9pm, Monday, 2021-11-22

作業規定與建議步驟：

1. 這個作業是要大家使用 React/Axios/Express 框架，寫出一個簡單的的全端應用 “Number Guessing Game”，也就是猜數字遊戲。
2. 這個作業沒有提供參考程式，但請參考 Lecture Note #5 中的說明以及參考程式，應該就可以簡單的完成 baseline 的要求。
3. **這個作業強制大家要使用 React/Axios/Express 框架，沒有按照此規定者，請 reviewer 將之評定為屍體。此外，也請按照講義針對 “package.json” 以及目錄/檔案名稱的建議來編寫程式。換句話說，請將前、後端程式分別放在 “frontend” 以及 “backend” 的目錄，然後在 “hw6” 目錄底下執行 “yarn server” 開啟 server (at Port 4000)，以及在另外一個 terminal, 也是在 “hw6” 目錄底下，用 “yarn start” 開啟猜數字的網頁 (at Port 3000)。**
4. “Number Guessing Game” 的基本要求功能如下：
   - (frontend/src/App.js) 處理畫面邏輯以及管理 states。
     - ✓ 當 “start game” 按鈕按下去之後，透過 axios 通知 server 去產生一個新的猜謎數字
     - ✓ 當 “guess!” 按鈕按下去之後，把輸入匡猜的數字透過 Axios 傳給 server 端去做判斷，並且接收 server 傳回來的猜測狀態，設定對應的 states 與 status 值 (字串)。Note: 數字是否符合格式或者是否在 1 ~ 100 之間，並不是在 frontend 處理，而是會傳到後端處理。當後端告知 error 時，應該要用 “catch” 來抓到 error, 並且印出像 ‘Error: "xx" is not a valid number (1 - 100)’ 這樣的 error message，然後要可以繼續猜先前未猜到的數字。
     - ✓ 當猜到數字後，要切換畫面顯示 "you won! the number was xxx.”, 然後當使用者按下 “restart” 按鈕後，透過 axios 通知 server 去重新產生一個新的猜謎數字，並且進入 “guess” mode.
   - (fronend/src/axios.js) 實現上述需要跟 server 溝通的 { startGame, guess, restart } 這三個 functions
   - (backend/core/getNumber.js) 定義一個全域變數 “number” (as an in-memory DB), 並且 export 兩個 functions: getNumber() 以及 genNumber(), 前者用來取得 “number”, 後者用來產生一個介於 1 ~ 100 的隨機亂數。
   - (backend/routes/guess.js) 實作 Express router-level middleware, 定義 { "/start", "/guess", "/restart" } 三個 APIs。其中 “/guess” 檢查猜的數字是否正確，並且回傳建議 (status, as a string) - “Bigger” (猜大一點), “Smaller” (猜小一點), “Equal" (答對了)，而如果輸入的數字格式不正確，請回傳的 HTTP error (406)；而 “/start" 與 “/restart" 兩個 APIs 是用來開始以及重新開始遊戲。
5. “Number Guessing Game” 的進階要求功能如下：
   - 在 “src/axios.js” 的各個 functions 處理 server not responding or not connected 的情況 (例如：server 被關掉)，並且在畫面 (src/App.js) 給予適當的錯誤碼 (i.e. HTTP 50x) 與錯誤訊息。(Note: 當 server 回來以後應該要可以繼續遊戲)
   - 把遊戲改成人類 (client 端) 與電腦 (server 端) 可以互相猜謎的遊戲，例如：1A1B, Tic-Tac-Toe, 等，最後可以判斷人類或者是電腦獲勝。Note: 由於 HTTP 的溝通是 “被動式” 的單向溝通，即只有當 client 端 request 時 server 才會回應，而無法由 server 主動發出 request, 因此，遊戲的設計應該是要 follow

此原則 — 當人類猜玩謎之後，電腦應該在判斷完之後，就立即決定是否要由電腦端繼續猜謎，並且將猜謎的結果回傳至 client 端。

- 任何 UI/UX 的改進 (請在 README.md) 中說明

6. 如果你 hw6 資料夾中有些檔案是沒有需要 push 到 GitHub 上去的，請記得把他們的檔名加到 .gitignore 去。
7. 以上說明，如有不夠詳盡之處，請自行判斷與定義，不用問我們「這樣可不可以」、「這個要不要做」之類的問題，再次聲明 practice homework 不會有詳細的規格，「定義規格」也是 practice 的一部分，但為了讓 reviewer 能夠正確地使用你的 app, 請儘可能地在 README.md 中提供必要的說明。

繳交

請在 deadline (9pm, Monday, 2021-11-22) 之前把你的作業 (i.e. hw6 整個目錄) push 到你 GitHub 上 wp1101 這個 repo 上面 (Note: 我們不接受遲交哦！)。你應該會用到的指令有 git add hw6 (i.e. 把整個 hw6 一次加進去), git commit -m “你想留的 comment”, 以及 git push. 你也可以用 git status 來檢查一下檔案是否都已經成功 push 到 GitHub 上面，或者是登入 GitHub 去檢查。請注意，你不用建立 ”review” 這個資料夾，助教會在指派 review 作業時再幫你加進去你的 repo。

如有任何的問題，請到 FB 社團的 "HW#6 Q&As” 貼文底下發問，或者是寫信到老師/助教信箱：eewebprogramming@googlegroups.com 。