VSCode: Open Project -> View Terminal -> npm run pull -> npm start

Eggsercises

Lecture 29

“Hit me with Your Best Shot”

Announcements

- All 3 parts of PS07 due Wednesday at midnight
- Quizzes 3 and 4 imported into My101
- Tutoring tonight 6-8 in FB007
- Review session Thursday 5-6 in FB009
- (Last) Quiz this Friday

Today:

Egg Russian Roulette

https://www.youtube.com/watch?v=vVhNc0

Overview

- One carton of 12 eggs
- 8 are hard boiled, 4 are raw
- The eggs are shuffled, randomized in carton
- Players alternate choosing an egg at random and crack it on their head
- First to select 2 raw eggs loses

Code Read: `28 – eggsercises`

- Play Rock-Paper-Scissors with a neighbor
- Winner reads and eggsplains `Egg.ts`
- Not-quite-winner eggsamines and discusses `Carton.ts`
- Check-in on `pollev.com/comp101unc` when you have a sense of what both `Egg.ts` and `Carton.ts` contain

Code Walk

- Let's review a few files together, then we're ready to begin
- `EggRoulette.ts` - where we'll implement the game logic
- `app-script.ts` - the starting point of our app
- `Graphics.ts` - the code for drawing the carton, eggs, and scoreboard
- First, let's reveal the content of the eggs while we're working on the game

Hands-on #1) Boiling the Eggs

- At `TODO #1.1`, in `EggRoulette.ts`, define a function named **`boilEggs`**
  1. It has a single parameter of type **`Carton`** and returns nothing
  2. It should loop through all but the last 4 elements of **`carton`**'s **`eggs`** array property
     - Why not loop through the last 4 elements?
  3. It should call the **`boil`** function on those Egg objects

- At `TODO #1.2`, in the constructor, call the **`boilEggs`** function using **`this.carton`** as the argument

- You should see 8 of the carton's Eggs become "hard boiled"

- Check-in when you think you've got it or are completely stuck.

```
// TODO #1.1: Define the function boilEggs

let boilEggs = (carton: Carton): void => {
  for (let i = 0; i < carton.eggs.length - 4; i++) {
    let egg = carton.eggs[i];
    boil(egg);
  }
};
```

Follow Along #2) Picking an Egg

- The **`pick`** function is called from the Graphics code each time you click an Egg. The parameter **`index`** refers to the index of the Egg clicked in Carton's Egg array.

- At `TODO #2`, declare a variable named **`egg`**, of type Egg, and assign it to the Egg at **`index`** in the **`carton`** property's **`eggs`** array. It will refer to the Egg object that was clicked

- **`if egg`** is **not cracked**, then
  1. crack it (by calling its crack function)
  2. `if` egg is raw, increment the property **`scores`** array at index **`currentPlayer(roulette)`** by 1
  3. increment the `turn` by 1

- The first time you click on any egg, score should go up by 1 if raw 0 if not. Second time you click on any egg, because it's "cracked", nothing happens. Check-in on PollEv!

```
export let pick = (roulette: EggRoulette, index: number): void => {
  // TODO #2: pick function - logic for picking an Egg
  let egg: Egg = roulette.carton.eggs[index];
  if (!egg.isCracked) {
    crack(egg);
    if (egg.isRaw) {
      roulette.scores[currentPlayer(roulette)]++;
    }
    roulette.turn++;
  }
};
```

Hands-on #3) Determining a Winner

- The first player to have a score of 2 (2 raw eggs picked) loses, so the other player is the winner

- In the **`winner`** function (`TODO #3`), implement the following logic:
  - `if` player 0's score (scores array index 0) is 2, then `return` (player) 1
  - Otherwise, if player 1's score (scores array index 1) is 2, then `return` (player) 0
  - Otherwise, `return` -1

- Now, when a player picks 2 raw eggs, the winner will show. Check-in when your conditional logic is working.

```
export let winner = (roulette: EggRoulette): number => {
  if (roulette.scores[0] === 2) {
    return 1;
  } else if (roulette.scores[1] === 2) {
    return 0;
  } else {
    return -1;
  }
};
```

Follow Along #4) Shuffling the Eggs

1. At `TODO 4.1`, define a function named **`shuffleEggs`**, with no parameters, that returns nothing

2. It should loop through every **`Egg`** in the **`carton`**'s array of **`eggs`**, using **`i`** as the index counter variable

3. In the loop, choose a random index and assign it to a local variable **`random`**
   - Generate a random index:
     `Math.floor(carton.eggs.length * Math.random())`

4. It should swap the Egg at index **`i`** with the Egg at index **`random`**
   - What's the key to swapping two values in memory? Hint: use a third variable

5. **Call this function in the constructor at `TODO 4.2`**. Save & refresh a few times. You should see your raw eggs repositioned at random.

- Back in **`app-script.ts`**, change **`revealEggState`** to be false and try playing your neighbor. Check-in when complete!

```
// TODO #4: Define the function shuffleEggs
let shuffleEggs = (carton: Carton): void => {
  let size: number = carton.eggs.length;
  for (let i: number = 0; i < size; i++) {
    let random: number = Math.floor(size * Math.random());
    let temp: Egg = carton.eggs[i];
    carton.eggs[i] = carton.eggs[random];
    carton.eggs[random] = temp;
  }
};
```

At Home:

- Read through the code from today's lecture
- Explain to yourself / a friend what is going on
- Make sure you ***understand*** what is going on in Egg.ts, Carton.ts, and EggRoulette.ts and how they all work together to create the game
- Trace through the code as you are playing the game in your browser
  - Can you match what is happening graphically to what is happening in your code?