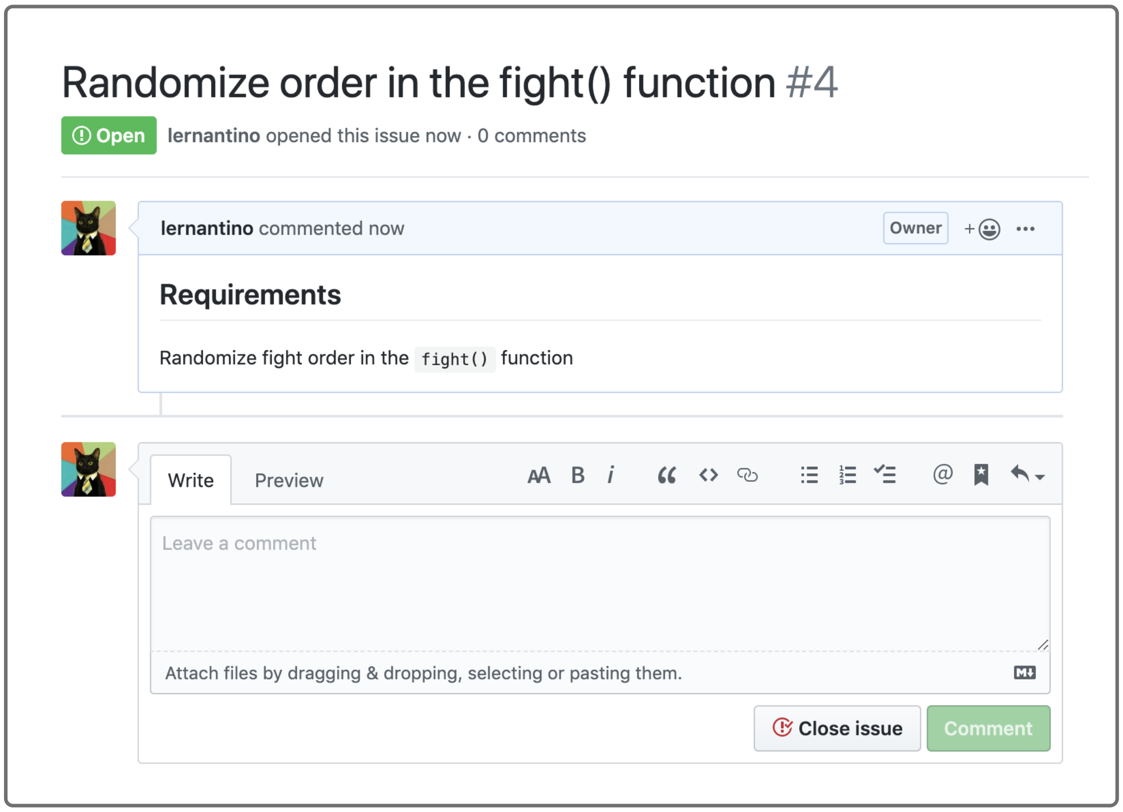
3.5.7 Randomize the Fight Order

To increase our gameplay appeal, let's randomize whether the user or the enemy robot can attack first at the beginning of each round. Take a look at the GitHub issue:



Randomize order in the fight() function #4

Open lernantino opened this issue now · 0 comments

lernantino commented now

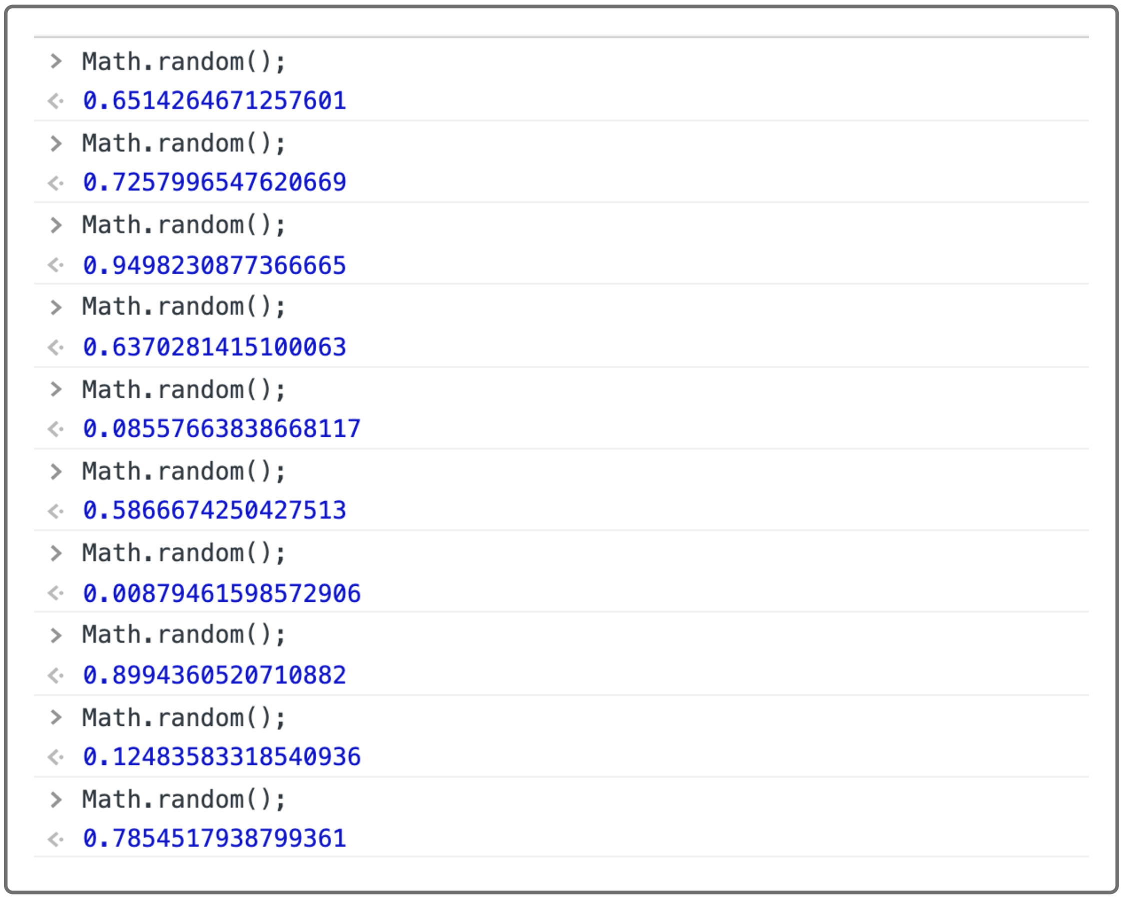
Requirements

Randomize fight order in the `fight()` function

Owner
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Close issue
Comment

Let's create a new feature branch called `feature/random-order`. We'll use our knowledge of the `Math` object and `random()` method to find a solution. `Math.random()` outputs a random decimal number between 0 and 1.

Type `Math.random()` several times into the console of the browser to see the results:



```
> Math.random();
< 0.6514264671257601
> Math.random();
< 0.7257996547620669
> Math.random();
< 0.9498230877366665
> Math.random();
< 0.6370281415100063
> Math.random();
< 0.08557663838668117
> Math.random();
< 0.5866674250427513
> Math.random();
< 0.00879461598572906
> Math.random();
< 0.8994360520710882
> Math.random();
< 0.12483583318540936
> Math.random();
< 0.7854517938799361
```

Notice that the `random()` method returns a decimal number between 0 and 1.

How can we use these numbers to simulate a coin flip?

We could use 0.5 as a halfway point and assign our coin's sides accordingly.

Here's an example of a conditional statement that will display heads and tails depending on the `Math.random()` output:

```
if (Math.random() > .5) {
  console.log("heads");
}
else {
  console.log("tails");
}
```

Let's use this conditional statement to assign the turn to the player or enemy robot. First let's declare the variable that will determine if it's the player robot's turn by initializing this value to true at the top of the `fight()` function:

```
// keep track of who goes first
var isPlayerTurn = true;
```

Next, use the `Math.random()` method to create a conditional statement that will reassign the `isPlayerTurn` to `false` 50% of the time. Put this `if` block below the `isPlayerTurn` variable initialization:

```
if (Math.random() > 0.5) {
  isPlayerTurn = false;
}
```

Now that we can successfully randomize the value of our turn variable, how do we implement this to affect the attack order of the robots? Let's use this Boolean value in the `while` loop to create conditional statements to execute attacks.

First let's pseudocode it:

- If it is the player robot's turn:
    - Prompt the fight or skip request

    - Remove damage from enemy robot's health
    - Check if the enemy robot has enough health to continue fighting
- If it is not the player robot's turn:
    - Prompt the fight or skip request
    - Remove damage from the player robot's health
    - Check if the player robot has enough health to continue fighting
- Switch turns for the next bout of fighting.

The last step is a critical one so that an exchange of fighting can occur.

On your own, use this pseudocode to modify the `fight()` function and randomize the fighting order. And remember—using Google to find the answers to any questions you have is always allowed and encouraged!

HIDE HINT

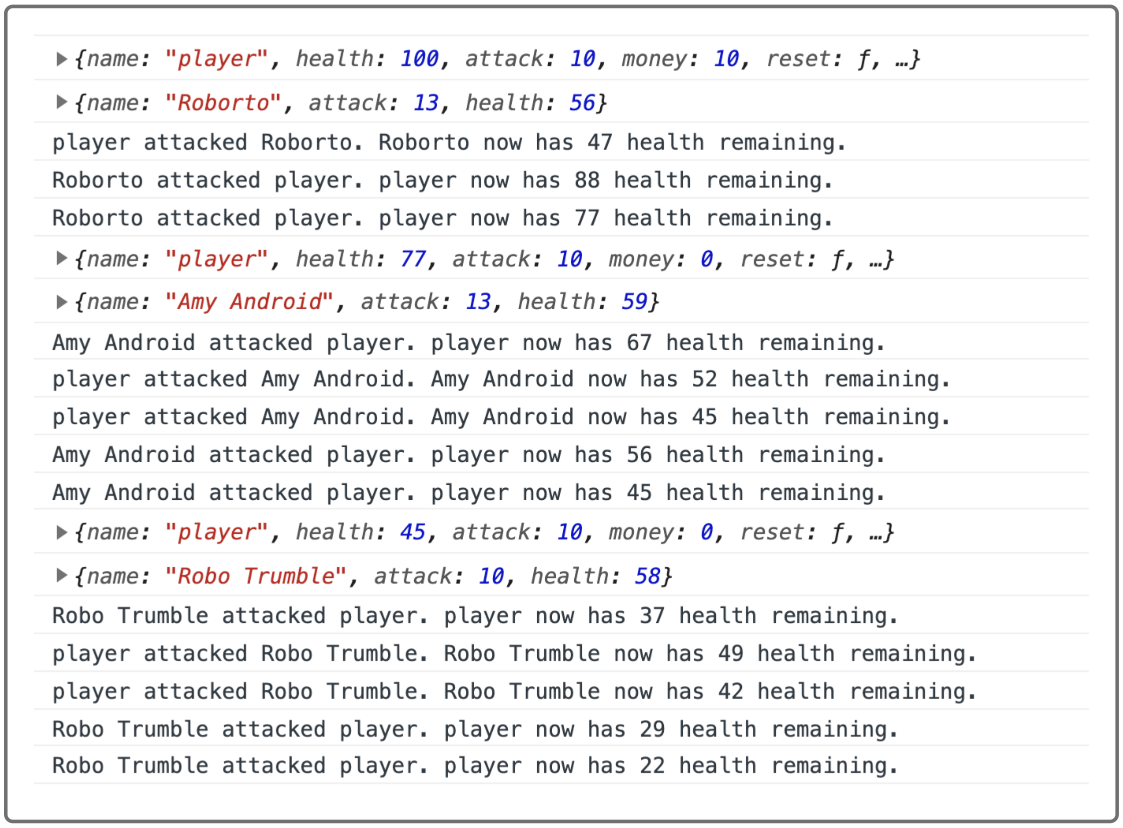
To switch turns, use the "!" operator to flip a Boolean value of the `isPlayerTurn`.

The modified `fight()` function should now look like this:

```
var fight = function(enemy) {
  // keep track of who goes first
  var isPlayerTurn = true;

  // randomly change turn order
  if (Math.random() > 0.5) {
    isPlayerTurn = false;
  }

  while (playerInfo.health > 0 && enemy.health > 0) {
    if (isPlayerTurn) {
      // ask user if they'd like to fight or skip using fightOrSkip fu
      if (fightOrSkip()) {
        // if true, leave fight by breaking loop
        break;
      }

      var damage = randomNumber(playerInfo.attack - 3, playerInfo.atta

      // remove enemy's health by subtracting the amount we set in the
      enemy.health = Math.max(0, enemy.health - damage);
      console.log(
        playerInfo.name +
          " attacked " +
          enemy.name +
          ". " +
          enemy.name +
          " now has " +
          enemy.health +
          " health remaining."
      );

      // check enemy's health
      if (enemy.health <= 0) {
        window.alert(enemy.name + " has died!");

        // award player money for winning
        playerInfo.money = playerInfo.money + 20;

        // leave while() loop since enemy is dead
        break;
      } else {
        window.alert(enemy.name + " still has " + enemy.health + " hea
      }
      // player gets attacked first
    } else {
      var damage = randomNumber(enemy.attack - 3, enemy.attack);

      // remove enemy's health by subtracting the amount we set in the
      playerInfo.health = Math.max(0, playerInfo.health - damage);
      console.log(
        enemy.name +
          " attacked " +
          playerInfo.name +
          ". " +
          playerInfo.name +
          " now has " +
          playerInfo.health +
          " health remaining."
      );

      // check player's health
      if (playerInfo.health <= 0) {
        window.alert(playerInfo.name + " has died!");
        // leave while() loop if player is dead
        break;
      } else {
        window.alert(playerInfo.name + " still has " + playerInfo.heal
      }
    }
    // switch turn order for next round
    isPlayerTurn = !isPlayerTurn;
  }
};
```

Notice how the "!", known as the **not operator**, is used in the value switching operation to reassign the `isPlayerTurn`. This step is critical for exchanging turns.

Let's test our new function to make sure our fight order has become randomized:



```
▶ {name: "player", health: 100, attack: 10, money: 10, reset: f, …}
▶ {name: "Roborto", attack: 13, health: 56}
player attacked Roborto. Roborto now has 47 health remaining.
Roborto attacked player. player now has 88 health remaining.
Roborto attacked player. player now has 77 health remaining.
▶ {name: "player", health: 77, attack: 10, money: 0, reset: f, …}
▶ {name: "Amy Android", attack: 13, health: 59}
Amy Android attacked player. player now has 67 health remaining.
player attacked Amy Android. Amy Android now has 52 health remaining.
player attacked Amy Android. Amy Android now has 45 health remaining.
Amy Android attacked player. player now has 56 health remaining.
Amy Android attacked player. player now has 45 health remaining.
▶ {name: "player", health: 45, attack: 10, money: 0, reset: f, …}
▶ {name: "Robo Trumble", attack: 10, health: 58}
Robo Trumble attacked player. player now has 37 health remaining.
player attacked Robo Trumble. Robo Trumble now has 49 health remaining.
player attacked Robo Trumble. Robo Trumble now has 42 health remaining.
Robo Trumble attacked player. player now has 29 health remaining.
Robo Trumble attacked player. player now has 22 health remaining.
```

As can be seen in the console, the first battle with each new robot has been randomized so now the advantage of the first attack is more balanced and fair.

And now we've reached that joyous moment when we can merge this new feature into `master` and start working on our next feature. Don't forget to close the issue as well.