CMPSC 302
WEB DEVELOPMENT

Review

- So far, our game has the following features:
  - A start button that begins the game when clicked
  - Buttons that:
    - When clicked, add to a list of user selections
    - Make silly beeps, boops, and other noises
- What we have left:
  - Playing turns
  - Pattern generation
  - Generated pattern vs. player-selected pattern checking
  - Earning player points
  - Losing the game
    - Because you will

Taking turns

- A “turn” consists of:
  - The program generating and displaying a pattern
    - “Replaying” that pattern so that the player knows what to click
  - The player clicking the pattern in the order offered
  - Program’s determination that the pattern chosen is correct
- This involves:
  - Writing code to generate patterns in response to the “level” a player is on
    - “Level” corresponds directly to the *number* of entries in a pattern
      - e.g. Level 2 is two clicks, Level 10 is 10
  - Displaying this pattern in a way that is understandable to the player, so that they can reproduce it to pass a “level”

Leveling up

- We know that a pattern consists of what outcomes?
  - Buttons:
    - top-left
    - top-right
    - bottom-left
    - bottom-right
- “Levels” are directly proportional to the number of steps in a pattern
- And, we must choose from these *at random* each time

Leveling up

```
// Get button ids in an array
let buttons = document.querySelectorAll(".game-button");
const elements = Array.from(buttons).map(button => {
        return button.id
});
```

Like forEach, but returns a result

Programming speak for "send back"

Leveling up

Generates a number between 0 and 1.00

Rounds result down

```
// Choose a random button
let choice = Math.floor(Math.random() * buttons.length);
// Represent this choice
console.log(buttons[choice]);
```

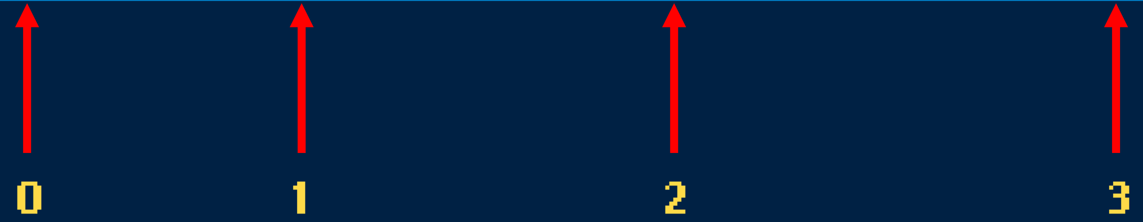
Because our array as 4 values, we can say that its "length" is 4

Selects the option corresponding to the number randomized

Learning to count…again

- We say the following array has 4 values:

['top-left', 'top-right', 'bottom-left', 'bottom-right']



- So, Math.random() * buttons.length equals 0 < x < 3
  - buttons[0] is actually the first item
  - buttons[3] is the last item

Putting it together

Create a variable that we can't change later

Which is a function that doesn't need any additional information

```
const chooseRandomBtn = () => {
  let idx = Math.floor(Math.random() * buttons.length);
  return buttons[idx].id;
};
```

...that sends back a single ID of a randomly-selected button each time I run it

Getting called out

- To use this function (chooseRandomBtn), we call it:

```
chooseRandomBtn();
```

A function call

Leveling up (cont'd)

- But, we need to do this how many times?
  - Enough equal to the "level" a player is on
- What are some ways we might do this with the knowledge we have?
  - Some assumptions:
    - As long as the player successfully chooses the right combinations, we're only adding one item each time
    - Could we store this in a global variable we can access anywhere?

Checking your work

```
const validatePattern = () => {
  if(choices.length !== pattern.length) return false;
  for(var i = 0; i < pattern.length; i++)
  {
    if(pattern[i] !== choices[i]) return false;
  }
  return true;
}
```

If the two patterns aren't the same length, then something's wrong!

false is a "Boolean" type: it's either "yes" or "no"

A different kind of for statement that checks every entry in both arrays to see if they're the same

If we've survived all of our tests, return that it's a valid match!