Initial app flow (pre-MVP)

* User enters site
* User sees homepage with 4 house colours and image of sorting hat in the middle
* When the user hovers over image, it appears larger, and a speech bubble is displayed saying “click me to begin the sorting process”
* When user clicks on the image, a question is displayed in the middle, with four answer options to choose from (displayed in four separate elements around the question) – this is displayed with its own styling (different to the homepage styling)
* When a user clicks on an answer:
  + the next question and relevant answer options are displayed in the exact same styling format
  + Depending on which answer is clicked, a variable for one of the four houses is increased by one
  + This means I need to somehow link a given answer option to a certain house variable
* This keeps repeating each time a user clicks on an answer
  + This means that each question and its four “answer choices” needs to be associated with a state variable – each time the state variable value changes, so does the question that is displayed.
* Once all questions have been worked through, the user sees the same page formatting as the homepage, but this time the sorting hat is telling the user which house they have been sorted into.

Problem of how to randomise order of answer choices whilst still maintaining their link to a particular “house”:

1. Data will be stored in questionsArray
2. questionsArray will be an array of objects, each pertaining to one question
3. Each object will contain the properties questionNumber (of type number), questionText (of type string), answerChoices (of type array)
4. answerChoices will be an array of strings with a length of 4
5. The answerChoices array will always be in the order gryffindor, slytherin, raveclaw, Hufflepuff
6. A “scores” array of objects (pertaining to each house) is also declared, with each object having a house property and score property, with all score values set to zero
7. To render the answer choices to the user:
   1. A function will get the answer choices array and randomise its order into a new array
   2. A map function will then render the list of items from the randomised array
8. When a user clicks on an answer choice
   1. An event handler function will listen for the click
   2. This event handler function should compare the text content of the clicked item to each string in the original answer choice array
   3. Whichever answer choice is the match, the appropriate value within the scores array should be updated by +1
   4. A tracker variable should also be updated to show which house’s “score” was last updated e.g. if the Gryffindor variable increased by one, the tracker variable should now be “gryffindor”
9. When a user clicks on the back arrow:
   1. The tracker variable value is checked
   2. The score variable for the house currently being tracked should be updated by -1
   3. The tracker variable should then be updated to what it was previously (will need a “history” array to keep tabs of what the tracker variable was previously)
10. When question becomes 11:
    1. The four house variables are compared
    2. If (there are two house variables of equal value that are also the largest value) the user is told they are a hat stall between these two houses
    3. Else the user is told they are sorted into whichever house has the largest value and:
       1. If (there are two or three house variables of equal value that are the second largest) the user is shown no secondary houses
       2. Else (if there is one house variable that has the second largest value) the user is told they have traits of this secondary house too

Plan for function to create a new array that randomises the current array, but keeps the current array intact:

* Declare shuffledArray as an empty array
* Check length of current array
* Iterate through each item in current array and:
  + Use Math.floor(Math.random()\*currentLength) to obtain an integer value between 0🡪length-1
  + Whilst an item in shuffledArray already exists at the index position of this randomly generated integer, keep regenerating this random integer
  + Place the item in current array at the randomly generated index position in shuffledArray