```
const BASE_API_URL = "https://jservice.io/api/";
const NUM_CATEGORIES = 6;
const NUM_CLUES_PER_CAT = 5;

// categories is the main data structure for the app; it looks like this:

//  [
//    { title: "Math",h
//      clues: [
//        {question: "2+2", answer: 4, showing: null},
//        {question: "1+1", answer: 2, showing: null}
//        ...
//      ],
//    },
//    { title: "Literature",
//      clues: [
//        {question: "Hamlet Author", answer: "Shakespeare", showing: null},
//        {question: "Bell Jar Author", answer: "Plath", showing: null},
//        ...
//      ],
//    },
//    ...
//  ]

let categories = [];


/** Get NUM_CATEGORIES random category from API.
 *
 * Returns array of category ids
 */

async function getCategoryIds() {
  // ask for 100 categories [most we can ask for], so we can pick random
  let response = await axios.get(`${BASE_API_URL}categories?count=100`);
  let catIds = response.data.map(c => c.id);
  return _.sampleSize(catIds, NUM_CATEGORIES);
}

/** Return object with data about a category:
 *
 *  Returns { title: "Math", clues: clue-array }
 *
 * Where clue-array is:
 *   [
 *      {question: "Hamlet Author", answer: "Shakespeare", showing: null},
 *      {question: "Bell Jar Author", answer: "Plath", showing: null},
 *      ...
 *   ]
 */

async function getCategory(catId) {
  let response = await axios.get(`${BASE_API_URL}category?id=${catId}`);
  let cat = response.data;
  let allClues = cat.clues;
  let randomClues = _.sampleSize(allClues, NUM_CLUES_PER_CAT);
  let clues = randomClues.map(c => ({
    question: c.question,
    answer: c.answer,
    showing: null,
  }));

  return { title: cat.title, clues };
}

/** Fill the HTML table#jeopardy with the categories & cells for questions.
```

```
 *
 * - The <thead> should be filled w/a <tr>, and a <td> for each category
 * - The <tbody> should be filled w/NUM_QUESTIONS_PER_CAT <tr>s,
 *   each with a question for each category in a <td>
 *   (initally, just show a "?" where the question/answer would go.)
 */

async function fillTable() {
  // Add row with headers for categories
  $("#jeopardy thead").empty();
  let $tr = $("<tr>");
  for (let catIdx = 0; catIdx < NUM_CATEGORIES; catIdx++) {
    $tr.append($("<th>").text(categories[catIdx].title));
  }
  $("#jeopardy thead").append($tr);

  // Add rows with questions for each category
  $("#jeopardy tbody").empty();
  for (let clueIdx = 0; clueIdx < NUM_CLUES_PER_CAT; clueIdx++) {
    let $tr = $("<tr>");
    for (let catIdx = 0; catIdx < NUM_CATEGORIES; catIdx++) {
      $tr.append($("<td>").attr("id", `${catIdx}-${clueIdx}`).text("?"));
    }
    $("#jeopardy tbody").append($tr);
  }
}

/** Handle clicking on a clue: show the question or answer.
 *
 * Uses .showing property on clue to determine what to show:
 * - if currently null, show question & set .showing to "question"
 * - if currently "question", show answer & set .showing to "answer"
 * - if currently "answer", ignore click
 * */

function handleClick(evt) {
  let id = evt.target.id;
  let [catId, clueId] = id.split("-");
  let clue = categories[catId].clues[clueId];

  let msg;

  if (!clue.showing) {
    msg = clue.question;
    clue.showing = "question";
  } else if (clue.showing === "question") {
    msg = clue.answer;
    clue.showing = "answer";
  } else {
    // already showing answer; ignore
    return
  }

  // Update text of cell
  $(`#${catId}-${clueId}`).html(msg);
}

/** Start game:
 *
 * - get random category Ids
 * - get data for each category
 * - create HTML table
 * */

async function setupAndStart() {
  let catIds = await getCategoryIds();
```

```
  categories = [];

  for (let catId of catIds) {
    categories.push(await getCategory(catId));
  }

  fillTable();
}

/** On click of restart button, restart game. */

$("#restart").on("click", setupAndStart);

/** On page load, setup and start & add event handler for clicking clues */

$(async function () {
    setupAndStart();
    $("#jeopardy").on("click", "td", handleClick);
  }
);
```