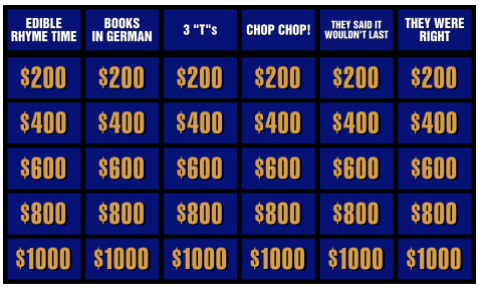
SER421: Web-based Programming — Client API Access Lab

The goal of this lab is to get your client-side apps to access a remote API. Keep in mind for Vue code that we are using the Vue framework version 3 with the Options API ONLY. PLEASE FOLLOW SUBMISSION INSTRUCTIONS!

Activity 1 (60 points): Play Jeopardy!

Jeopardy! is a popular trivia game show that has been on US television almost continuously in some form since 1964. Standard gameplay as show on TV is in 3 rounds, 2 using the Jeopardy "board" and one for Final Jeopardy. An example board is shown below.



The categories are given along the top, and dollar values are given. 3 players play in a given game. One player chooses a category and a dollar value, and an answer is revealed. In Jeopardy, answers are given, and the player must respond in the form of a question. For example, a player might request "They Were Right for $400" and the answer revealed is "This man is never wrong" to which the player should respond "Who is Dr. Gary?" to earn the $400 😊 . In the real game the first of the players to "buzz in" gets to attempt a question to the answer, and if incorrect, the other players may attempt. A wrong attempt results in the player's dollar balance being deducted by the amount of the selection, a correct answer results in it being added. We will modify the rules a bit in our version for practical purposes.

Requirements:

R1. Implement a game with 3 players and a board that has *4* categories by *5* questions per category. There are a number of ways to implement the game board; for example, an HTML canvas or an HTML table, or a CSS way of setting up the table. This is your choice. The initial board should display categories along the top, and dollar values in $100 increments down the columns.
   a. The gameboard will be populated by questions from the Open Trivia Database (opentdb.com).
      i. Your application should randomly select 4 categories from the API's available categories.
      ii. Each category (column) should use Easy questions for dollar values $100 and $200, Medium questions for $300 and $400, and a Hard question for $500. More on the API below.

R2. When a player clicks on a dollar value, the app should show what category and value was selected followed by a trivia question displayed below the table in true-false format.

R3. The player then answers the trivia question by selecting either True or False choices.
   a. If the player answers correctly, then:
      i. A "`Correct`!" message is displayed just below where the question was.
      ii. The "box" for the just completed question should show in **green:** "**`P#`**" where # is the player's number.
      iii. The dollar amount is added to that player's balance.
      iv. The player selects the next category/dollar value.
   b. If the player answers incorrectly, then:
      i. An "`Incorrect`!" message is displayed just below where the question was
      ii. The "box" for the just completed question should show in **red:** "**`P#`**" where # is the player's number
      iii. The dollar amount is deducted to that player's balance
      iv. Control for the next selection passes to the next player.
   c. Play continues until all questions on the board have been completed. When the board is completed, a message is displayed below the board regarding the winner, e.g. "`Player 3 has won the game`!"

Here is what the page would look like during hypothetical gameplay where Player 1 just answered a question correctly:

`Player 1: $500`	`Player 2: -$700`	`Player 3: $200`	
`I love Javascript!`	**`Class is cancelled`**	**`Famous professors`**	**`Not your Mom's PERLs`**
`P3`	**`$100`**	**`P1`**	**`$100`**
`$200`	**`P2`**	**`$200`**	**`$200`**
`$300`	**`$300`**	**`$300`**	**`P3`**
`$400`	**`$400`**	**`$400`**	**`P1`**
`$500`	**`$500`**	**`P2`**	**`$500`**

Note this is a mockup in Word; you have flexibility in rendering the board as long as it is a 4x5 game board.

```
Player 1 selects Famous Professors for $100:
Dr. Gary is your Program Chair:  o True  ●False
Correct!
Player 1 to select
```

Constraints:

C1. You must implement this solution in Vue (not VanillaJS). Deciding when to do the fetch calls is a big part of the assignment.
C2. External libraries are not needed nor allowed. If you feel you need to use a library to do something, clear it with Dr. Gary first.

Submission:

You are allowed to choose whether to implement a single-file (SFC) solution or a multiple-file (Vue3 build project) solution as a zipfile. Name your file labfetch_act1_<asurite>.[vue|zip]. As always you may have a README.[md|txt] to convey to us any information you'd like us to know before we test your application.

Activity 2 (40 points): Enhance your Jeopardy Game

Activity 2 asks you to make some enhancements to Activity 1. Please keep the submission (and your code locally) separate for the two submissions, we will not grade one submission against both Activity specifications!

Requirements:

R5. Double Jeopardy is an event where the player is invited to "wager" an amount of money up to the amount they have in her/his current balance, or the value from the board, whichever is larger. Augment requirement R2 such that:
   a. Double Jeopardy happens roughly 10% of the time (or roughly twice per board) based on a random number generator.
   b. If Double Jeopardy happens,
      i. If the player's balance is less than the face value on the board of the question, use the face value
      ii. Else prompt the player to provide an amount to wager, e.g. "Double Jeopardy wager: ___" and check s/he has not entered a value greater than her/his balance.
      iii. Use this value when adjusting the score in requirement R3.

R6. The game should support a configurable number of players and categories, but always use 5 questions per category as before. To dynamically configure the game:
   a. At the top of the page should be a prompt: "Number of players: " followed by a small textbox or dropdown that accepts a number in the range of 2-6.
   b. Underneath should be similar a prompt for "Number of categories: "
   c. When values are given for the prompts in a and b, then the resulting player scoreboard and gameboard should be populated as in R1.a, except the number of questions will be dynamic based on the number of categories.

Submission:

Instructions are the same as Activity 1, just name your file labfetch_act2_<asurite>.[vue|zip]. Make sure you make a separate submission zipfile, we will not score one solution for both activities.

The OpenTDB API:

You should read the OpenTDB API documentation to understand how to use the API. The API provides ways to get questions by category, get only True/False questions, and get questions of varying difficulty (Easy, Medium, Hard). There is a nice GUI interface to show you how to construct URLs to make API calls.

Your application should use the API at https://opentdb.com/api_category.php to get the list of available categories (there are presently 24), and randomly select 4 to start a game. *Note: prior experience with the API shows there are no medium difficulty Boolean questions for category 13, and no hard difficulty questions for categories 21, 27, 30, or 32. I am fine with filtering out these categories and only allowing API calls on the other 19 categories.* Beyond that, how many API calls and when you make them to administer gameplay is up to your design.

Example: Suppose one of your categories is "Science: Computers". You could populate the $100 and $200 questions in a property by accessing https://opentdb.com/api.php?amount=2&category=18&difficulty=easy&type=boolean and getting result:


```
{
  "response_code": 0,
  "results": [
    {
      "category": "Science: Computers",
      "type": "boolean",
      "difficulty": "easy",
      "question": "RAM stands for Random Access Memory.",
      "correct_answer": "True",
      "incorrect_answers": [
        "False"
      ]
    },
    {
      "category": "Science: Computers",
      "type": "boolean",
      "difficulty": "easy",
      "question": "The NVidia GTX 1080 gets its name because it can only render at a 1920x1080 screen resolution.",
      "correct_answer": "False",
      "incorrect_answers": [
        "True"
      ]
    }
  ]
}
```