2018

Technical University of Moldova
Faculty of Computers, Informatics and Microelectronics

Laboratory Work Nr. 1

WEB TECHNOLOGIES AND APPLICATIONS

STUDENT: NICOLAE SÎRBU, GR. TI-171M
TEACHER: VLADIMIR PODDUKIN, LECT. SUP.

ToDo:
- Think of a Quiz application that can be developed using JavaScript;
- Define functional requirements;
- Define use cases;
- Develop the application.

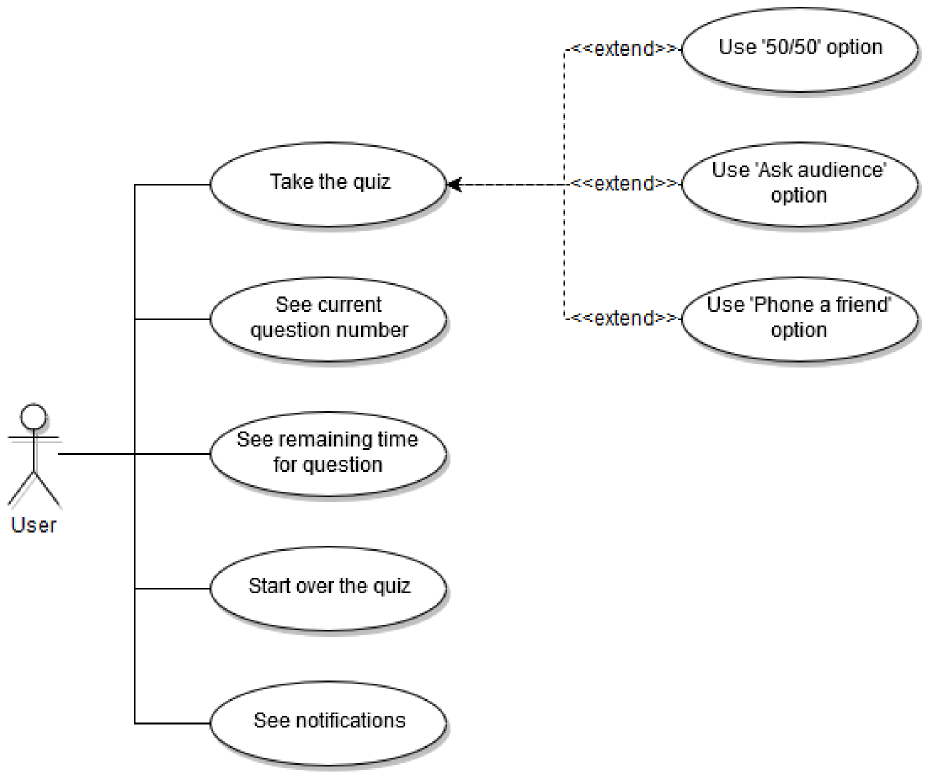
The application that I'm going to build in this laboratory work represents a quiz of 10 questions about JavaScript programming language.

Functional requirements

- As a user, I want to take a quiz, so that I could know how good is my knowledge about JavaScript.
- As a user, I want to see the current question number, so that I could see the progress of the quiz.
- As a user, I want to see a timer, so that I could be aware of the remaining time for the current question.
- As a user, I want to have three lifelines: „50/50", „Ask audience" and „Phone a friend", so that I could use some help while passing the quiz.
- As a user, I want to have a start button, so that I could start the quiz over and over again.
- As a user, I want to see some notifications, so that I could be aware if I have passed or not the quiz.

Use cases

The functional requirements defined above are represented in the following Use Case diagram:



User
Take the quiz
See current question number
See remaining time for question
Start over the quiz
See notifications
<<extend>>
<<extend>>
<<extend>>
Use '50/50' option
Use 'Ask audience' option
Use 'Phone a friend' option

Fig. 1 Use Case diagram

Let's have a closer look to the Use Cases defined in the diagram above:

1. *Take the quiz* – this is basically the main functionality of this application, answer to some questions by choosing one of the four available options.
2. *See current question number* – the user must see some indicator on the page that would display the current level, the current question, so that the user could be aware about the progress he has made.
3. *See remaining time for question* – every round consists of a question. Once it has been answered, the user goes to the next level. But, in order to make this application a bit more challenging, a countdown

has been added to each level, so that is the user doesn't manage to answer the question within the given time, he loses the "game". For this purpose, the timer must be displayed somewhere on the page.

4. *Start over the quiz* – there must be a button that would allow to the user to start again the quiz.
5. *See notifications* – the user must see some notifications when something important happened, like he won or lost the "game".

While passing the quiz, the user can choose one of the helper options, that would ease his path to the victory:

- *Use '50/50' option* – disables 2 answers that are not correct, thus letting the user to choose from the two remaining, one of which is wrong, and another one is correct.
- *Use 'Ask audience' option* – generates some random percentage for every possible answer. The correct answer always gets the biggest percentage.
- *Use 'Phone a friend' option* – simulates a call to a friend which always gives you the right answer.

Source code

The questions for this quiz are kept inside an array in a dedicated file, called *questions.js*. The data within the array has a JSON format:

```
/**
 * This array represents the data source of the quiz.
 */
var questions = [
    {
        questionTextValue: "To find highest number in an array, method to be used is",
        level: 1,
        answers: [
            {
                id: 1,
                answerTextValue: "Math.highest",
                isCorrect: false
            },
            {
                id: 2,
                answerTextValue: "Math.max()",
                isCorrect: true
            },
            {
                id: 3,
                answerTextValue: "Math.largest",
                isCorrect: false
            },
            {
                id: 4,
                answerTextValue: "Math.cal(high)",
                isCorrect: false
            }
        ]
    }
]
```

Every Question is an object, which has three properties:

- *questionTextValue* – the text of the question;
- *level* – the level for which this questions belongs;
- *answers* – an array of answers.

The third property in its turn is also an object with another three properties:

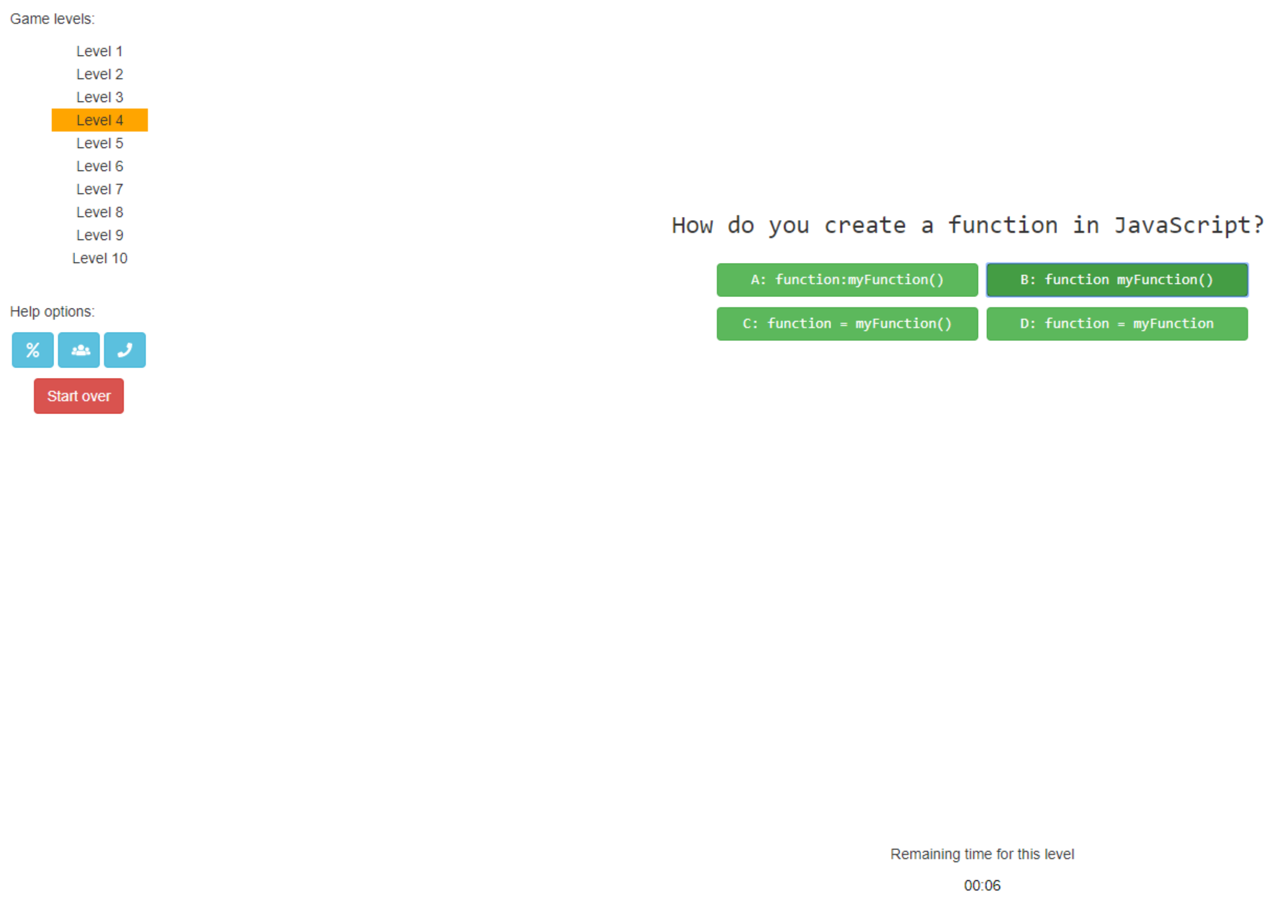
- *id* – the ID of the answer;
- *answerTextValue* – the text of the answer;
- *isCorrect* – indicates if the answer is correct or not.

This quiz has been built using AngularJS. The main file that contains all the logic for this quiz is called *quiz_controller.js*. The source code presented below is just a part of the controller. As you can see, I used two types of functions, function declarations – the functions that are used within this controller only and function expressions – the functions that interact with the view.

```
var MAX_LEVELS = 10;
var TIME_PER_LEVEL = 11; // seconds
var countdown;

/**
 * This is the first method that is called when Angular initializes the component.
 */
$scope.init = function () {
    $scope.isQuizStarted = false;
};

/**
 * This method represents the entry point for starting the quiz.
 */
$scope.startQuiz = function () {
    $scope.isQuizStarted = true;
    $scope.currentLevel = 1;
    $scope.currentQuestion = getRandomQuestionForLevel($scope.currentLevel);
    resetLevelIndicator();
    highlightCurrentLevel($scope.currentLevel);
    startCountdown();
    enableHelpOptionButtons();
};

/**
 * Gets a random question for the given level from the array of questions.
 * @param level - the level for which to get a question
 */
function getRandomQuestionForLevel(level) {
    var questionsForLevel = getQuestionsForLevel(level);
    var randomIndex = Math.floor((Math.random() * questionsForLevel.length));
    return questionsForLevel[randomIndex];
}

/**
 * Gets all the questions for the given level from the array of questions.
 * @param level - the level for which to get the questions
 */
function getQuestionsForLevel(level) {
    return $window.questions.filter(function (question) {
        return question.level === level;
    });
}

/**
 * Validates the answer the user has chosen by clicking on one of the four buttons.
 * @param answerIndex - the index of the button on which the user has clicked on
 */
$scope.validateAnswer = function (answerIndex) {
    if ($scope.currentQuestion.answers[answerIndex].isCorrect) {
        goToNextLevel();
    } else {
        endQuiz('You lost. \nWould you like to start a new quiz?');
    }
};

/**
 * Passes to the next level by loading a new question, highlighting the appropriate level and
restarting the
 * countdown.
 */
function goToNextLevel() {
    $scope.currentLevel++;

    if ($scope.currentLevel <= MAX_LEVELS) {
        highlightCurrentLevel($scope.currentLevel);
        $scope.currentQuestion = getRandomQuestionForLevel($scope.currentLevel);
        startCountdown();
    } else {
        endQuiz('Congratulations, you won. \nWould you like to start a new quiz?');
    }
}
```

```
/**
 * Handles the click on 50/50 helper option.
 */
$scope.useFiftyFiftyOption = function () {
    var wrongAnswers = getWrongAnswersForQuestion($scope.currentQuestion);
    var possibleCombinations = getAnswersCombinationsToDisable(wrongAnswers);
    var randomCombinationIndex = Math.floor((Math.random() * 3));
    var firstAnswerToDisable = possibleCombinations[randomCombinationIndex].charAt(0);
    var secondAnswerToDisable =
possibleCombinations[randomCombinationIndex].charAt(possibleCombinations[randomCombinationIndex].length
- 1);

    $('#answer_' + firstAnswerToDisable).prop('disabled', true);
    $('#answer_' + secondAnswerToDisable).prop('disabled', true);
    $('#fifty_fifty').prop('disabled', true);
};
```

Quiz screenshots



How do you create a function in JavaScript?

A: function:myFunction()
B: function myFunction()
C: function = myFunction()
D: function = myFunction

Remaining time for this level

00:06

Fig. 2 Main view of the quiz

Conclusion

I this laboratory work I built a small web application that acts as a quiz offering to the user a possibility to test his knowledge in terms of JavaScript. It has some well-known options like a timer indicating the remaining time for a question, the current question number, and some helper options that might give some hints to the user while answering the questions.

The project has been created using AngularJS, but the focus has been oriented on JavaScript mostly, as the main task of this laboratory work was to study and use the basics of JavaScript.