Project 3 - Online Quiz

Aim:

To build an online quiz where users can answer a series of questions and receive feed-back or graded on their performance

Overview

The online quiz would be a website where users can answer a series of questions and receive feedback or graded on their performance. The quiz would have questions with multiple-choice options, and navigation buttons (like 'Next' and 'Submit'). Each question could be displayed on its own page or section. The quiz could feature different types of questions (like multiple-choice, true/false, or fill-in-the-blank).
The questions should be loaded at random every time. The online quiz demonstrates your understanding and application of JavaScript, along with HTML and CSS for structure and styling. At the end of the quiz, users could receive a summary of their performance, including the number of correct answers, total score, and possibly explanations for the correct answers.

Key Elements

- **Quiz Theme:** You have the freedom to choose the subject or theme of your quiz. It can range from general knowledge to specific topics like science, literature, or pop culture.

- **How To Page:** Include a dedicated page with clear instructions on how to take the quiz. This page should guide users on starting the quiz, navigating through questions, and understanding how their answers will be evaluated.

- **JavaScript Functionality:** Utilize JavaScript effectively to handle the quiz mechanics. This includes question navigation, capturing user responses, calculating scores, and displaying results. Your JavaScript code should be clean, well-commented, and efficient.

Assessment Criteria

Usability Impact

- Design a web application that meets accessibility guidelines, follows the principles of UX design and presents a structured layout and navigation model, and meets its given purpose.
- Design the organisation of information on the page following the principles of user experience design (headers are used to convey structure, information is easy to find due to being presented and categorised in terms of priority).
- Write custom JavaScript, HTML and CSS code to create a responsive front-end web application consisting of one or more HTML pages with significant interactive functionality.
- Implement an interactive web application that incorporates images or graphics of usable resolution, consistent styling, undistracted foregrounds.
- If used, implement clear navigation to allow users to find resources on the site intuitively.

Layout and Visual Impact

- Design interactivity for a web application that lets the user initiate and control actions and gives feedback.
- Design a web application that meets accessibility guidelines (e.g., contrast between background and foreground colours to cater for the visually impaired) Optionally, add alt text for non-text elements.
- Include graphics that are consistent in style and colour.
- Ensure that foreground information is never distracted by backgrounds.
- Ensure that you use up to a maximum of three colours in the web design.

Code Quality

- Write custom HTML code that passes through the official W3C validator with no issues.
- Write custom CSS code that passes through the official (Jigsaw) validator with no issues.
- Write JavaScript code that passes through a linter (e.g., Jshint) with no significant issues.
- Implement appropriate working functionality for all project requirements.
- Ensure that there are no broken internal links or sections.
- Deploy a final version of the code to a cloud-based hosting platform (e.g., GitHub Pages) and test to ensure it matches the development version.
- Use CSS media queries across the application to ensure the layout changes appropriately and maintains the page's structural integrity across device screen sizes.

Code Quality

- Organise HTML, CSS, and JavaScript code into well-defined and commented sections.
- Clearly separate and identify code written for the website and code from external sources (e.g., libraries or tutorials).
- Attribute all code from external sources to its original source via comments above the code and (for larger dependencies) in the README.

- Organise code and assets files in directories by file type.
- Group files in directories by file type (e.g., an assets directory will contain all static files and may be organised into sub-directories such as CSS, images, etc.)
- Write a README.md file for the web application that explains its purpose, the value that it provides to its users, and the deployment procedure.
- Name files consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility.
- Insert screenshots of the project features, give a brief description of what each feature does and explain its value to the user.
- Use Git & GitHub for version control of an interactive web application up to deployment.

Getting Started

In this section, it covers the necessary resources and requirements that are needed to complete the project and ensure that you satisfy all requirements.

Session with your Mentor:

Before commencing the project, you will need to book a session with your Mentor to plan how you will go about completing the project, understanding the project brief, planning, and staging, and finally how to go about submitting the project.

Documentation:

Write a README.md file for your project that explains what the project does and the value that it provides to its users. Attribute any code from other sources. Document all testing procedures and findings.

Version Control:

Use Git & GitHub for version control in making regular commits to your GitHub repo.

- Make sure you are adding, committing, and pushing your code regularly throughout creating your project.
- When committing your code to GitHub, ensure you commit messages describing what code you have changed since the last commit.

Deployment:

Deploy the final version of your code to a hosting platform such as GitHub Pages. You can follow the steps below to deploy it:

- Open your project repository on GitHub, Go to the ‘Settings’ tab.
- Scroll down to ‘GitHub Pages’
- Under ‘Source’, click on the ‘None’ dropdown and choose ‘Master’.
- Make sure the Root Folder is selected and click ‘Save’.
- Your page will reload, scroll back down to GitHub Pages and you will now see a URL. This is a link to a live version of your site.

Testing:

Make sure to run your code through code validation & quality tools for example:

- HTML - https://validator.w3.org/#validate_by_input
- CSS - http://jigsaw.w3.org/css-validator/
- JavaScript - https://javascriptvalidator.net/
- Document all errors and warnings found, make sure you document your process of fixing these errors.

Attribution:

Maintain clear separation between code written by you and code from external sources using clear comments. Include the URL of where you found the code (e.g., libraries or tutorials).

Resources

Main Technologies Required: HTML, CSS

Optional: Bootstrap and/or other CSS libraries/frameworks/any other JavaScript libraries like jQuery.

GitHub Documentation: https://er-bharat1992.medium.com/writing-readme-md-markdown-file-file-bd711d1afbfa

GitHub Commits: https://github.com/sw1ckham/i-brary/commits/master

Readme Project example: https://github.com/Allwrightben/starquiz

Mentor Support: Via your dedicated mentor