DOCUMENTATION FOR QUIZ-APP REPOSITORY

jumana07.github.io

JUMANA HASEEN Frontend Developer

Table of Contents

Overview of Quiz-App File Documentation

<u>index.html</u>

HTML File Documentation

This document provides documentation for the given HTML file. Let's go through each section of the HTML code to understand its purpose and functionality.

Document Type Declaration

The first line of the HTML code is the Document Type Declaration. It specifies the version of HTML being used and ensures that the browser renders the page correctly. In this case, the doctype is `DOCTYPE html>`, which indicates that the HTML5 standard is being used.

HTML Structure

The `<html>` element is the root element of an HTML page. It encapsulates the entire content of the page.

Head Section

The `<head>` section contains metadata and other information about the HTML document. It is not visible on the webpage itself. Inside the `<head>` section, we have the following elements:

- `<meta charset="UTF-8">`: This specifies the character encoding for the document, ensuring that special characters and symbols are displayed correctly.

- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: This meta tag sets the viewport properties, which helps in rendering the webpage appropriately on different devices.

- `<title>`: This element sets the title of the webpage, which is displayed in the title bar of the browser.

- `<link rel="stylesheet" href="style.css">`: This `<link>` element is used to link an external CSS file called "style.css" to the HTML document.

- `<script defer src="script.js"></script>`: This `<script>` element is used to link an external JavaScript file called "script.js" to the HTML document. The `defer` attribute ensures that the script is executed after the HTML document has been parsed.

Body Section

The `<body>` section contains the visible content of the webpage. It includes the following elements:

- `<header>`: This element represents the header section of the webpage. Inside the `<header>` element, there is an `<h1>` heading with the text "Quiz Quest".

- `<main>`: This element represents the main content of the webpage. Inside the `<main>` element, there is a `<form>` element.

- `<form action="Questions.html">`: This `<form>` element is used to create a form for user input. The `action` attribute specifies the URL to which the form data will be submitted.

- `<label for="file">`: This `<label>` element is used to create a label for the file input field.

- `<input type="file" id="csvFile" accept=".csv">`: This `<input>` element is of type "file" and allows the user to select a file from their device. The `accept` attribute restricts the file selection to only files with the ".csv" extension.

Button

After the `<main>` section, there is a `<button>` element. This button serves as a submit button for the form. It has the following attributes and functionality:

- `type="submit"`: This attribute specifies that the button is a submit button, which submits the form data.

- `value="start"`: This attribute sets the value of the button to "start".

- `name="startbtn"`: This attribute sets the name of the button to "startbtn".

- `id="uploadbtn"`: This attribute sets the ID of the button to "uploadbtn".

- `class="mainbtn"`: This attribute sets the class of the button to "mainbtn".

- `onclick="sendFile()"`: This attribute specifies a JavaScript function called "sendFile()" to be executed when the button is clicked.

Conclusion

This documentation provides an overview of the structure and functionality of the given HTML file. It explains the purpose of each section and element within the HTML code, helping developers understand and work with the file more effectively.

script.js

JavaScript File Documentation

This documentation provides an overview of the JavaScript file provided. The file contains code that handles the uploading and processing of a CSV file.

Variables

- `uploadedfile`: This variable is used to store the uploaded file object.

- `csv_data`: This variable is used to store the data from the uploaded CSV file.

Event Listener

The code snippet includes an event listener that listens for a change event on an input element. When a file is selected for upload, the event is triggered, and the code inside the event listener is executed.

Uploading and Processing the File

Within the event listener, the selected file object is assigned to the `uploadedfile` variable. A `FileReader` object is then created to read the contents of the file.

The `readAsBinaryString` method of the `FileReader` object is used to read the file as a binary string. Once the file is successfully read, the `onload` event is triggered with the file contents stored in the `event.target.result` property.

The file contents are then assigned to the `csv_data` variable, and a message containing the contents is logged to the console.

Saving File Data

The `sendFile` function is defined to save the `csv_data` to the local storage. The `localStorage.setItem` method is used to store the data with the key "fileData".

Please note that the code provided assumes that the HTML file includes an input element for file uploads. Make sure to include the necessary HTML markup to utilize this JavaScript code properly.

style.css

CSS File Documentation

This documentation provides an overview of the CSS file and its different sections. It explains the purpose and usage of each CSS rule and selector.

Global Styles

The global styles section defines styles that are applied to all elements on the webpage. These styles include:

- `padding: 0;` - Sets the padding of all elements to 0.

- `margin: 0;` - Sets the margin of all elements to 0.

- `box-sizing: border-box;` - Sets the box-sizing property of all elements to border-box, which includes padding and border within the element's total width and height.

Custom CSS Variables

The :root selector is used to define custom CSS variables that can be used throughout the stylesheet. These variables are defined using the -- prefix and can be assigned any valid CSS value. The variables defined in this CSS file include:

- `--btncolor: #ffe74cff;` - Defines the color for buttons.

- `--wrong: #ff5964ff;` - Defines a color for indicating a wrong answer.

- `--white: #ffffffff;` - Defines the color white.

- `--correct: #6bf178ff;` - Defines a color for indicating a correct answer.

- `--bgcolor: #35a7ffff;` - Defines the background color for the webpage.

- `--textbg: #D9D9D9;` - Defines the background color for text elements.

- `--fontfam: Arial, Helvetica, sans-serif;` - Defines the font family for the webpage.

Body Styles

The body styles section defines styles that are applied to the `<body>` element. These styles include:

- `background: url("./images/bg.png");` - Sets the background image of the webpage using the specified URL.

- `background-repeat: no-repeat;` - Prevents the background image from repeating.

- `background-color: var(--bgcolor);` - Sets the background color of the webpage using the custom CSS variable --bgcolor.

- `background-size: cover;` - Scales the background image to cover the entire background area.

- `display: grid;` - Specifies that the body element should be displayed as a grid container.

- `grid-template-rows: 1fr;` - Sets the grid template rows to one fractional unit, which allows the content to take up the available vertical space evenly.

- `place-items: center;` - Centers the grid items both horizontally and vertically.

- `gap: 6rem;` - Sets the gap between grid items to 6rem.

- `font-family: var(--fontfam);` - Sets the font family of the webpage using the custom CSS variable --fontfam.

Heading Styles

The h1 styles section defines styles that are applied to `<h1>` elements. These styles include:

- `font-size: clamp(3px, 6rem, 8rem);` - Sets the font size of the heading element to a value between 3px and 8rem, based on the available space.

- `text-align: center;` - Centers the text within the heading element.

- `color: var(--white);` - Sets the text color of the heading element using the custom CSS variable --white.

Input File Styles

The #inputfile styles section defines styles that are applied to the element with the id "inputfile". These styles include:

- `padding: 1rem;` - Sets the padding of the element to 1rem.

- `background-color: var(--btncolor);` - Sets the background color of the element using the custom CSS variable --btncolor.

- `background-image: url("./images/wordupload.jpg");` - Sets the background image of the element using the specified URL.

- `background-position: right;` - Positions the background image to the right.

- `background-size: 50px;` - Sets the size of the background image to 50px.

- `background-repeat: no-repeat;` - Prevents the background image from repeating.

- `border-radius: 1rem;` - Sets the border radius of the element to 1rem.

Label Styles

The label styles section defines styles that are applied to `<label>` elements. These styles include:

- `font-weight: bold;` - Sets the font weight of the label element to bold.

- `font-size: clamp(0.5rem, 2rem, 5rem);` - Sets the font size of the label element to a value between 0.5rem and 5rem, based on the available space.

Main Button Styles

The .mainbtn styles section defines styles that are applied to elements with the class "mainbtn". These styles include:

- `background-color: var(--btncolor);` - Sets the background color of the elements using the custom CSS variable --btncolor.

- `width: clamp(2rem, 5rem, 10rem);` - Sets the width of the elements to a value between 2rem and 10rem, based on the available space.

- `height: 3rem;` - Sets the height of the elements to 3rem.

- `border-radius: 1rem;` - Sets the border radius of the elements to 1rem.

Main Container Styles

The main container styles section defines styles that are applied to the `<main>` element. These styles include:

- `container: main inline-size;` - Specifies that the `<main>` element should be a container with an inline size.

Media Queries

The CSS file includes media queries that target specific screen sizes and orientations. These media queries define additional styles for different devices. The media queries in this CSS file include:

- Media query for screens with a width less than 480px and in portrait orientation:

- Sets various styles for all elements on the webpage to ensure optimal display on small screens.

- Adjusts the font size, width, and other properties to fit the smaller screen size.

- Media query for screens with a width between 481px and 768px and in landscape orientation:

- Sets various styles for all elements on the webpage to ensure optimal display on medium-sized screens in landscape orientation.

- Adjusts the font size, width, and other properties to fit the screen size and orientation.

Please note that the specific values used in the media queries may vary depending on the design requirements of the webpage.

Questions.html

HTML Document Documentation

This documentation provides an overview and explanation of the structure and elements used in the given HTML file.

Document Type Declaration

The HTML file begins with the Document Type Declaration, which is specified using the `DOCTYPE html>` tag. This declaration tells the web browser that the document is written in HTML5.

HTML Element

The HTML element is the root element of the HTML file. It encloses all other elements and represents the entire HTML document. In the given file, the HTML element has the attribute `lang` set to "en" to specify the language of the document as English.

Head Element

The Head element contains metadata and other information about the HTML document. It is enclosed within the `<head>` tags. In the given file, the Head element includes the following elements:

- Meta Charset: This meta element specifies the character encoding for the HTML document. The `charset` attribute is set to "UTF-8" to indicate the UTF-8 character encoding.

- Meta Viewport: This meta element defines the viewport settings for the HTML document. It ensures that the document is displayed properly on different devices and screen sizes. The `viewport` attribute is set to `width=device-width, initial-scale=1.0`.

- Title: The title element specifies the title of the HTML document, which is displayed in the browser's title bar or tab. In the given file, the title is set to "Quiz-Questions".

External Stylesheets

The HTML file includes two external stylesheets using the `<link>` element. These stylesheets are used to apply CSS styles to the HTML content. The `rel` attribute is set to "stylesheet" to indicate that the linked file is a stylesheet. The `href` attribute specifies the path to the CSS file to be linked. In the given file, two stylesheets are linked: "quizstyle.css" and "style.css".

Body Element

The Body element represents the main content of the HTML document and is enclosed within the `<body>` tags. In the given file, the Body element contains the following elements:

- Heading: The `<h1>` element is used to define a heading. In this case, it displays the text "Quiz Quest".

- Quiz Container: The `<div>` element with the id "quiz-container" is a container that is used to hold the quiz content. It is empty in the given file and may be populated dynamically using JavaScript.

- JavaScript: The `<script>` element is used to embed or reference JavaScript code in an HTML document. The `defer` attribute is used to ensure that the script is executed after the HTML content has been parsed. The `src` attribute specifies the path to the JavaScript file to be included. In the given file, the script "script1.js" is referenced.

- Button Container: The `<div>` element with the id "btn-div" is a container that holds buttons for user interaction. It contains two buttons:

- Next Button: The first button is of type "button" and has an `onclick` attribute that triggers the function "showResult()". It also has the class "mainbtn" for styling purposes.

- Reset Button: The second button is also of type "button" and has an `onclick` attribute that triggers the function "resetPage()". It also has the class "mainbtn" for styling purposes.

This concludes the documentation for the given HTML file.

script1.js

JavaScript File Documentation

This JavaScript file is responsible for handling a quiz functionality. It retrieves data from the local storage, parses it into questions, answer options, and correct answers, and dynamically creates the necessary elements to display the quiz on a web page.
Variables
The JavaScript file declares and initializes the following variables:
1. `correct`: This variable keeps track of the number of correctly answered questions.
2. `wrong`: This variable keeps track of the number of incorrectly answered questions.
3. `lastClickTime`: This variable stores the timestamp of the last click event.
4. `file`: This variable retrieves the data stored in the local storage.
5. `lines`: This variable splits the retrieved data into an array of lines based on the newline character ('\n').
Creating Quiz Elements
The JavaScript file uses a `for` loop to iterate over each line of the `lines` array. For each line, it separates the line into questions, answer options, and correct answers. Then, it dynamically creates HTML elements to represent the question and answer options.
1. Question Element: A `<div>` element with the class "questions" is created. Inside this element, a `<p>` element is added to display the question text.
2. Option Element: An `<ul>` element is created to hold the answer options. For each answer option, an `<li>` element is created and appended to the `<ul>` element. Inside each `<li>` element, an `<input>` element of type "radio" and a corresponding label are added.
Appending Quiz Elements
The question element and option element created for each line of the `lines` array are appended to the quiz container. The quiz container is obtained using the `getElementById` method, targeting the element with the ID "quiz-container".
Event Listener and Result Calculation
The JavaScript file adds an event listener to a "Next" button. When the button is clicked, the `showResult` function is called. This function calculates the result of the quiz by checking the selected answer for

each question against the correct answer. The number of correct and wrong answers is updated accordingly.

Double-Click Prevention

To prevent counting double-clicks as separate clicks, the `showResult` function checks the time elapsed since the last click. If the time is less than 300 milliseconds, it is considered a double-click, and the event is prevented from further processing.

Opening Next Page and Resetting Quiz

After calculating the quiz result, the `openNext` function is called. This function stores the number of correct and wrong answers in the local storage and opens a new page called "result.html".

Additionally, the JavaScript file defines a `resetPage` function that clears the selection of all radio buttons on the quiz page. This function is not currently being called in the provided code.

quizstyle.css

CSS File Description

This CSS file contains styles for a web page. Let's go through each section to understand what each rule does.

Universal Selector

The first rule applies to all elements in the web page. It sets the margin and padding to 0 and box-sizing to border-box. This ensures that all elements have consistent spacing and sizing.

```css
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}
```

Body Styles

The second rule targets the `body` element. It sets the display property to flex and the flex-direction to column. This means that the content inside the body will be displayed in a single column.

```css
body {
display: flex;
flex-direction: column;
}
```

Questions Styles

The next rule targets elements with the class `questions`. It sets the display to block, padding to 0.3rem, margin to 2rem, and text-align to center. Additionally, it sets the background color to #D9D9D9, giving the elements a gray background.

```css

.questions {

display: block;

padding: 0.3rem;

margin: 2rem;

text-align: center;

background-color: #D9D9D9;

}

```

Empty Questions

The rule `questions:empty` targets elements with the class `questions` that are empty. It sets the display property to none, making the empty questions invisible.

```css

.questions:empty {

display: none;

}

```

Unordered List Styles

The next rule targets unordered lists (`ul`) and removes the default list style.

```css

ul {

list-style-type: none;

}

```

List Item Styles

The rule targeting `li` elements sets the display property to inline-flex, align-content to space-between, margin to 2rem, padding to 2rem, and text-align to center. It also sets the font size using the `clamp()` function, which ensures that the font size is between 0.2rem and 2rem. The background color is set to #D9D9D9, and the font weight is set to bold.

```css

li {

display: inline-flex;

align-content: space-between;

margin: 2rem;

padding: 2rem;

text-align: center;

font-size: clamp(0.2rem, 1rem, 2rem);
background-color: #D9D9D9;

font-weight: bold;

}
```

Radio Input Styles

The rule targeting `input[type=radio]` sets the width of radio inputs to 2rem.

```css
input[type=radio] {

width: 2rem;

}
```

Button Container Styles

The rule targeting `btn-div` sets the text-align property to center. This is likely used to center any buttons within the container with the id `btn-div`.

```css

btn-div {

text-align: center;

}
```

Media Query

The last section of the CSS file is a media query. It applies styles to elements when the screen width is less than 480px and the orientation is portrait.

```css

@media only screen and (width < 480px) and (orientation=portrait) {

* {

min-width: 70vw;

font-size: 5px;

overflow: scroll;

word-wrap: break-word;

}

div {

width: 80vw;

}

li {

display: inline-block;

}

}
```

Within this media query, all elements (`*`) are given a minimum width of 70vw, font size of 5px, overflow set to scroll, and word wrap set to break-word. This ensures that the content is readable and appropriately sized on small screens. Additionally, `div` elements within the media query are given a width of 80vw, and `li` elements are displayed as inline-block.

<u>result.html</u>

HTML file Documentation

This is a simple HTML page that displays the result of a quiz. The page includes some styling using CSS and also utilizes JavaScript to dynamically update the result.

HTML Structure

The HTML structure of the page consists of the following elements:

- `DOCTYPE html>`: Specifies the document type as HTML.

- `<html lang="en">`: The root element of the HTML document, with the language attribute set to "en" for English.

- `<head>`: Contains meta information and the page title.

- `<meta charset="UTF-8">`: Specifies the character encoding for the document as UTF-8.

- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Sets the viewport width and initial scale for responsive design.

- `<title>Quiz-Resulttitle>`: Sets the title of the page to "Quiz-Result".

- `<link rel="stylesheet" href="style.css">`: Links an external CSS file named "style.css".

- `<style>`: Contains inline CSS styles specific to this page.

- `<h1>Quiz-Resulth1>`: The main heading of the page.

- `<div id="result">`: A container for the result buttons.

- `<button type="button" id="correct-btn">Correct Answerbutton>`: A button to display the number of correct answers.

- `<button type="button" id="wrong-btn">Wrong Answerbutton>`: A button to display the number of wrong answers.

- `<h3>Thank Youh3>`: A message to thank the user.

- `<script type="text/javascript">`: Contains JavaScript code to dynamically update the result.

- `var correct = localStorage.getItem('corr');`: Retrieves the number of correct answers from local storage.

- `var wrong  = localStorage.getItem('wro');`: Retrieves the number of wrong answers from local storage.

- `const corr_btn = document.getElementById("correct-btn");`: Gets the correct answer button element by its ID.

- `const wro_btn = document.getElementById("wrong-btn");`: Gets the wrong answer button element by its ID.

- `const span1 = document.createElement('span');`: Creates a new span element.

- `const span2 = document.createElement('span');`: Creates another span element.

- `span1.classList.add('rescorrect');`: Adds the CSS class 'rescorrect' to the first span element.

- `span2.classList.add('reswrong');`: Adds the CSS class 'reswrong' to the second span element.

- `span1.innerHTML = correct;`: Sets the inner HTML of the first span element to the number of correct answers.

- `span2.innerHTML = wrong;`: Sets the inner HTML of the second span element to the number of wrong answers.

- `corr_btn.appendChild(span1);`: Appends the first span element to the correct answer button.

- `wro_btn.appendChild(span2);`: Appends the second span element to the wrong answer button.

CSS Styling

The CSS styles in the `<style>` section define the appearance of various elements on the page. Some of the styles include:

- `h2`: Styles the heading 2 elements with font size, text alignment, and font weight.

- `result`: Styles the container div for the result buttons with text alignment.

- `.rescorrect` and `.reswrong`: Styles the correct and wrong answer buttons with font size, font weight, color, float, line height, and border.

- `button`: Styles all buttons with width, height, background color, font size, padding, margin, text alignment, and font weight.

- `h3`: Styles the heading 3 elements with font size, font weight, color, and margin top.

- `@media only screen and (width < 480px) and (orientation = portrait)`: Applies specific styles for screens with a width less than 480px and portrait orientation. Styles include adjusting the width and text alignment of the container div, modifying the size and font of the answer buttons, and centering the heading 3 element.

Overall, the CSS styles contribute to the visual design and responsiveness of the page, making it more user-friendly and aesthetically pleasing.