**TMA 03**

1a.

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>My Erehwon Diary ZX924023</title>

    <meta name="author" content="Stephen Rice" />

    <!-- Set viewport to ensure this page scales correctly on mobile devices -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link rel="stylesheet" href="tma03.css" />

    <!-- Set demo data -->

    <script src="tma03-demo.js"></script>

    <!-- Start TMA03 application -->

    <script src="tma03.js"></script>

  </head>

  <body>

    <h1>My Erehwon Diary ZX924023</h1>

    <main>

      <section id="text" class="button">

        <button type="button">Add entry</button>

      </section>

      <section id="image" class="button">

        <button type="button">Add photo</button>

        <input type="file" accept="image/\*" />

      </section>

    </main>

  </body>

</html>

1b.

(i)

section.entry textarea {

    /\* Use a variable-width font \*/

    font-family: sans-serif;

    /\* Make text areas fill their container \*/

    background-color: transparent;

    border-width: 3px;

    width: 100%;

    height: 100%;

    resize: none;

    /\* To make sizing easier, include borders or padding in width \*/

    box-sizing: border-box;

    /\* Allow scrolling using keyboard only \*/

    overflow: hidden;

}

Graphical user interface, text, application, email

Description automatically generated

(ii)

main {

    /\* Apply style rules to the diary container here? \*/

    display: flex;

    flex-wrap: wrap;

}

section {

    /\* To make sizing easier, include borders or padding in width \*/

    box-sizing: border-box;

    /\* Set a limit on entry widths? \*/

    flex-shrink: 1;

    flex-grow: 1;

    margin: 0.5rem;

}

Using flex because it’s easy to manipulate and is very flexible in its container. It is widely used and has lots of support available as well as being supported in most browsers. Flex box can resize to fit in its container which is great for different sized images and varying string lengths. Flex-box can be very I chose to use this approach because it seemed the easiest to implement and maintain given the potential scaling of the app.

Graphical user interface, application, PowerPoint

Description automatically generatedGraphical user interface, application

Description automatically generated

1c.

(i)

/\*\*

 \* Create and store demonstration items

 \*/

function createDemoItems() {

    console.log("Adding demonstration items to local storage");

    var item, data, key;

    // Make a demo text item

    data =

        "Friday: We arrived to this wonderful guesthouse after a pleasant journey " +

        "and were made most welcome by the proprietor, Mike. Looking forward to " +

        "exploring the area tomorrow.";

    item = makeDiaryItem("text", data);

    // Make a key using a fixed timestamp

    key = "diary" + "1536771000001";

    // Store the item in local storage

    //localStorage.setItem(key, item);

    // Make a demo text item

    data =

        "Saturday: After a super breakfast, we took advantage of one of the many " +

        "signed walks nearby. For some of the journey this followed the path of a " +

        "stream to a charming village.";

    item = makeDiaryItem("text", data);

    // Make a key using a fixed timestamp

    key = "diary" + "1536771000002";

    // Store the item in local storage

    //localStorage.setItem(key, item);

    // Make a demo image item

    data = window.DUMMY\_DATA\_URL;

    item = makeDiaryItem("image", data);

    // Make a key using a fixed timestamp

    key = "diary" + "1536771000003";

    // Store the item in local storage

    //localStorage.setItem(key, item);

    // Make a demo text item

    data =

        "Sunday: Following a tip from Mike we drove to a gastropub at the head of " +

        "the valley - a great meal and fabulous views all round.";

    item = makeDiaryItem("text", data);

    // Make a key using a fixed timestamp

    key = "diary" + "1536771000004";

    // Store the item in local storage

    //localStorage.setItem(key, item);

}

(ii)

    // Create an event listener to delete the entry

    function deleteEntry() {

        // A new version of this function is created every time addSection is called,

        // so it can access all the variables available in this call to addSection

        console.log("deleteEntry called with variables in scope:", {

            itemKey,

            childElement,

            sectionElement,

            deleteButton,

        });

        // Remove the section from the page

        sectionElement.parentNode.removeChild(sectionElement);

        // Remove the entry from local storage

        localStorage.removeItem(itemKey);

    }

(iii)

    // Create an event listener to save the entry when it changes

    // (i.e. when the user types into the textarea)

    function saveEntry() {

        // A new version of this function is created every time addTextEntry is called,

        // so it can access all the variables available in this call to addTextEntry

        console.log("saveEntry called with variables in scope:", {

            itemKey,

            initialText,

            isNewEntry,

            textElement,

        });

        // Q1(c)(iii) Task 1 of 2

        // ...get the textarea element's current value

        initialText = textElement.value;

        // ...make a text item using the value

       var diaryItem = makeDiaryItem("text", initialText);

        // ...store the item in local storage using the given key

        localStorage.setItem(itemKey, diaryItem);

    }

    // Q1(c)(iii) Task 2 of 2

    // Connect the saveEntry event listener to the textarea element 'change' event

    textElement.addEventListener("change", saveEntry);

}

(iv)

        // Q1(c)(iv) Task 1 of 2

        // Make an image item using the given url

        var diaryItem = makeDiaryItem("image", dataUrl);

        // Store the item in local storage using the given key

        localStorage.setItem(itemKey, diaryItem);

    // Add a 'dummy' image entry

    //addImage(window.DUMMY\_DATA\_URL);

    // Q1(c)(iv) Task 2 of 2

    // Get a reference to the file input, the target of the change event

    var inputElement = changeEvent.target;

    // Get the first selected file from the file input's files array

    // (if this is a multiple file input there will be more items in this array)

    var file = inputElement.files[0];

    function loaded(event) {

        // Get a "data URL" (a long string of text encoding the binary data)

        // from the load event target...

        var data = event.target.result;

        addImage(data);

    }

    // Make a FileReader to read the file

    var reader = new FileReader();

    // Attach a listener that will add our image

    reader.addEventListener("load", loaded);

    // Start reading the file

    reader.readAsDataURL(file);

    // Clear the file selection (allows selecting the same file again)

    changeEvent.target.value = "";

}

1d.

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Part 2.

**Prototype Report**

This report addresses all questions and concerns put forward by the client. It will also address any issues that may not have previously been considered. By the end of the report the objective is that the client has the confidence to move forward from the prototype on to a finished and useable development.

Prototype vs Finished Application – 500 word

The first thing we will be exploring in the prototype Guest Book Application are the limitations on the WEBAPI that is used to store diary entries on the application namely local-storage. Using the localStorage API gives access to the storage object. It is here that our prototype app is storing our entries and this is where we would start to see complications with the finished application. Using the local storage lets saved data persist across browser sessions so to a single user for example when testing in the prototype version the experience seems pleasant and all the entries they have given are there even if they close it and come back later. The issue here is because the entries are stored *locally* they are NOT available to any other user accessing the application from their own device, they will only see their own entries stored in their local storage which makes the prototype fundamentally unsuitable to use as a finished product alone. What we ought to be doing is storing all this information on a server where each user is access the server. However there are further issues to discuss, for example the local storage has a limited capacity for data. Once we reach the limit we will no longer be able to add entries or images to the application. Another point rather to the same effect is that as we discussed earlier the storage is persistent, enough so that it has no expiration date so maxing out on storage is a likely conclusion if we are not properly managing and removing items from local storage as necessary. Some other issues that are likely to be encountered where we to continue with the prototype are the clearing of cookies on the page will remove all current entries stored in local storage and any user on an ‘incognito’ or private browser will lose all of their entries as soon as the sessions ends.

Already Present

Upload Images

Add text entries

Edit Text entries

Display entries on the web page

Send data to server

Delete text/image entries

* Add text entries
  + This is the ability to add strings of text to the app
* Upload images
  + This is the functionality enabling the user to upload images to the app
* Display entries on the page
  + Text and image entries and displayed in formatted containers on the page
* Edit text entries
  + The ability to change an entry after submission
* Delete text/image entries
  + The ability to delete text or image entries after submission
* Send data to server
  + After submission of an entry end data to the server to be available for all users

Privacy and security – 500 words

Moving on from functionality we will look further into the privacy and security considerations for Mikes web application. One of the first privacy and security issues developers working on Mikes Guest Book web app will consider is the use of cookies. If Mike plans to harvest any data from his users then there will need to be built in consent for the user under the GDPR legislation. If we were to continue with the protype application and using local storage we would also see concerns here of the security of the data that has been posted. The information stored here can be manipulation and deleted by a user so should not be used a place to store any potentially sensitive data. We have also seen that Mike intends to use device sensors to track things such as geolocation. Again if we are to obtain a user’s location through there personal device we will need to consider how we will obtain permission to do so and how the data we retrieve will be stored securely. In terms of the security of Mike application we would want to authenticate a user, this is because perhaps Mike has competitors and they become aware of his new application. Ultimately without authenticating the user anyone could access the app and post slanderous reviews for their own gain. Mike could even become the victim of a cyber attack whereby the application if flooded with requests ultimately taking down the application. However with authentication also comes the responsibility of correctly securing transactions and user information, it is known that developers have user cookies to store information about authenticated users, as we know cookies can be read and manipulated leavers users vulnerable. This has brough us back to cookies. Some of the steps developers might take to secure this kind of information are using session cookies which are deleted once the session ends. Another option is to set a secure flag, the secure flag ensures a browser only sends cookie over the HTTPS connections.

In conclusion we have looked at issues facing the prototype application and considered what the impact of such things would be on a finished version. We have highlighted the main components of the application and moved on to then discuss any potential privacy and security concerns.

Post-mortem

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Issue Description | Resolution | Feed-forward |
| 1. | Arranging the diary entries using CSS | The resolution I can to after trying each implementation was to “Keep it simple” so I went with what I found easiest to implement, maintain and what is probably the most supported. | Try not to over engineer anything. Most often the simplest solution is the one we need. “Keep is simple”. |
| 2. | Adding photo entries to the diary using the FileReader | After spending too long trying to figure this out the solution was moving the call to the addImage method into the call back function. | This was more of a trial and error situation as nor warnings were produced for me to follow. Using log statements helped me to identify null and undefined values which led to the solution. I couldn’t use a variable outside of the callback function even though it was declared in the global scope. |
| 3. | TIME | Got it done the best I could. Around other modules and assignments. | Spend more time reading the material and not just picking it up when its time for an assignment. |
| 4. | Report writing | Same as above | Although I enjoy the practical element I need to spend some time understanding the concepts. |

Word Count: 1094

Mozilla (2022) Window.localStorage [Online] Available at: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (Accessed: 08 March 2022).

Open University (2022) 5.3 Limitations and security issues [Online] Available at: https://learn2.open.ac.uk/mod/oucontent/view.php?id=1881394&section=5.3 (Accessed: 08 March 2022).