**Discovering History**

**Product Manual**

**Alexus A. Fry**

**CS 6375 - Fall 2023**

**St. Mary’s University**

Contents

[Background and Overview 3](#_Toc166023969)

[Requirements 5](#_Toc166023970)

[Design 7](#_Toc166023971)

[Installation 14](#_Toc166023972)

[Using the System 15](#_Toc166023973)

[Sample Session 16](#_Toc166023974)

[Modifications 17](#_Toc166023975)

[Troubleshooting 18](#_Toc166023976)

[References 19](#_Toc166023977)

[Appendix A 20](#_Toc166023978)

# Background and Overview

Timelines are a way to organize information in a chronological order. This was not created by a single individual but rather evolved over time as a natural way of recording history. The concept of a timeline has evolved over time differing slightly. One of the oldest records of a timeline is the “Sumerian King List” depicting the length of the reigns of each king. It was inscripted in stone tablets.

A group of stone tablets with writing

Description automatically generated

During the Middle Ages, scrolls were used to document events and Monasteries were a big provider in the care, preservation, and creation of the documents by hand. Moving on to the invention of the printing press revolutionized the distribution of information and timelines became more widespread through manuscript, and books. The enlightenment era followed an increase interest in history and historical research in future works. This allowed timelines to be seen in a more graphical representation. With the creation of computers, timelines have become a more flexible topic in organizing and presenting historical information. For example, social media platforms use timelines to display users’ posts and activities in a chronological order. Timelines have evolved from simple written lists to sophisticated tools used for various purposes not limited to education or historical research but also visual story telling. They are a great tool to understand the passage of time.

Commonly used for historical documentation and education, they are invaluable in project management. It can be used to track tasks, milestones, and deadlines. Gantt charts, a type of timeline, are especially popular to utilize. IN journalism, new stories or biographies are presented in a visually compelling way through timelines. When going into family history, family trees, a timeline that maps out brth, marriage, and death of family member are used to visually represent a genealogy tree. Timelines are used in legal cases to present sequences of events to help understand complex cases. It can be used for personal planning to map out life events and future plans to help aid in short term and long-term goals. Timelines are also utilized to convey a story visually through art or as a narrative such as books.

As you can guess from the examples, timelines have evolved to be utilized more as a concept than a factual physical tool. There is a systematic approach to many of the examples. For the sake of Robert Gagne’s Nine Events of instructions, a framework used in education design, lets brainstorm how a timeline could potentially fulfill his events of instruction. For the sake of the project “Discovering History” we will be assuming the timeline is utilized as a website.

* **Gain attention**: Timelines themselves have a history of being more engaging than a simple display of information as timelines guide the user through its chronological nature.
* **Inform learners of the objective**: Inherently timelines do not have an objective, however, the website can set the expectations from exploring the timeline before the timeline it utilized. Usually, timelines are titled for simplicity and to inform users of the content though.
* **Stimulate recall of prior learning:** When reading further into an event provided by the timeline, allowing the user to answer a question about it after they’ve read further into the topic.
* **Present the content**: Information is accessed through an interactive timeline that is easy to follow.
* **Provide learning guidance**: When answering a prompted question, if incorrect, explanations, definition, or clarifications can be used to help learners understand the presented content. Sending them back to the information with a hint where to find the answer would fit this as well.
* **Elicit performance**: Incorporating a question about the information prompts users to engage with the content.
* **Provide feedback**: Following the users actions and answers will allow the website to hold incorrect questions for users to go back to help reinforce correct understandings.
* **Assess performance**: Allowing the user to go back to questions also allows users to calculate certain statistics of correct/incorrect questions, time spent on a topic, and so forth.
* **Enhancing retention and transfer**: Users could be prompted after answering a question that summarizes key information about it.

Attempting to structure a timeline website to follow Gagne’s Nine levels of instruction to make it a productive and effective educational website. A mockup design is presented below fulfilling most if not all his events of instructions.

# Requirements

Originally, I wanted to explore and share amusing stories from history without a specific theme, for light-hearted learning. When I reflected, history is often portrayed in a timeline for easy digestion, along with giving context, it can help reveal funny coincidences if you pay attention. History isn’t limited to one group of people, so understanding that this topic could extend to a wide range of people needs to be considered as I would want anyone with an interest to be able to access it. Being as my original goal is to learn, I would need to present information in a very straightforward and easy to follow manner. Ensuring the design is universal while following Gagne’s Instructional Principles, does make it antiquated, but building from the bottom up is never easy.

**Minimum Hardware and Software Requirements:**

**User:**

* **Hardware:**
  + Any device capable of running a modern web browser.
* **Software:**
  + A modern web browser (such as Chrome, Firefox, Safari, Edge, etc.).

**Developer:**

* **Hardware:**
  + Any device capable of running a modern web browser.
  + A computer for development.
* **Software:**
  + Text editor or integrated development environment (IDE) for writing HTML, CSS, and JavaScript code.
  + Web browser for testing and debugging.

**Goals/Purposes for the User:**

**Learn:** Users can learn about various events in computer history through articles displayed in the timeline.

**Review:** Users can review their understanding by answering questions related to the articles.

**Track Progress:** Users can track their learning progress and performance through profile stats.

**Product Functionalities**:

* **Timeline Display:** Display a timeline of events in computer history.
* **Article Navigation:** Allow users to navigate to articles corresponding to events by clicking buttons on the timeline.
* **Article Display:** Display detailed articles about historical events.
* **Review Section:** Provide a section where users can review their understanding of articles.
* **Question Display:** Display questions related to articles in the review section.
* **Profile Tracking:** Track the user's progress and performance, including the number of correct and incorrect answers.
* **Modal Windows:** Use modal windows to display profile stats and help information.
* **Toggle Visibility:** Allow users to toggle the visibility of certain elements, such as hiding and displaying the help section.

By implementing these elements and functionalities, users can effectively learn about computer history, review their understanding, and track their progress.

# Design

HTML, JavaScript (JS), and CSS are excellent choices for creating a website as they have extensive documentation and an active community. Finding support went without a hitch, when I encountered challenges during development. Each language is universally supported by web browsers, ensuring minimal compatibility issues and access to a wide audience.

Languages:

**HTML**: Fundamental web development language.

**JavaScript**: Allows user engagement through dynamic behavior such as the interactive timeline, pop-up modal, and quiz functionalities.

**CSS**: Allows customization of the website appearance to create appealing UI elements.

**Database Definitions:**

**InnerText: Count**

Stored answers from previously answered questions in option <button>’s

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| Right Answers | VARCHAR(5) | Number of correctly answered questions |
| Wrong Asnwers | VARCHAR(5) | Number of wrongly answered questions |

Example Data:

| **Name** | **Count** |
| --- | --- |
| Right Answers | 1 |
| Wrong Answers | 3 |

**Button: Events**

Stored information about historical events in timeline <button>’s

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| Description | VARCHAR(255) | Year of the event & short description of the event |
| ArticleLink | VARCHAR(255) | Link to the full article |

Example Data:

| **Description** | **ArticleLink** |
| --- | --- |
| 1947 Bug | iframe.html#bug |
| 1957 Hard Drive | Ifram.html#harddrive |
| ... | ... |

**Array: Questions**

Stored questions in relation to the specific event.

| **id** | **question** | **op1** | **op2** | **op3** | **Correct**  **Answer**  **Index** |
| --- | --- | --- | --- | --- | --- |
| 1 | What is the primary function of an antivirus software? | A. Protecting against viruses | B. Speeding up the computer | C. Deleting files | 1 |
| 2 | Which of the following is NOT a common way to get a virus? | A. Opening suspicious email attachments | B. Installing antivirus software | C. Downloading files from untrusted sources | 2 |
| ... | ... | ... | ... | ... | ... |

Each row in the table corresponds to a single question.

* **ID**: unquie identifier for each question
* **Question:** Text of the question
* **Op1,Op2,Op3**: Mulitple-Choice answers for the question
* **Correct Answer Index**: the correct answer among the options.

This structure allows you to store and retrieve each question and its corresponding options and correct answer from the database efficiently. This can be further efficient if you combine all arrays into one to reduce redundancy and storage.

**Major Code Modules:**

1. **HTML Structure (Home.html & iframe.html)**:
   * Description: Defines the structure of the web page, including the timeline, modal windows, buttons, and content areas.
2. **CSS Styling (style.css)**:
   * Description: Defines the styles and layouts for various elements on the web page, including colors, fonts, spacing, and responsiveness.
3. **JavaScript Logic (script.js & iscript.js)**:
   * Description: Contains the client-side logic for handling user interactions, such as displaying modals, loading event data, handling quiz functionality, and updating the UI dynamically.

**System Flow**

The user interacts with the system by first seeing Welcome Modal, which exiting will bring to the Home Screen. The Home screen contains four elements: two buttons and two columns. One button, the Statistics, displays the current sessions answer tally. The other button, Help, holds a brief description of the website and functionality. Scrolling the right column allows the user to explore events at their own pace. Selecting an event to the right will make the second column go to a correlated detailed article. These are the primary ways of navigation. When a user finishes an article, there will be a series of questions at the end to test the users knowledge. Users’ stats will update accordingly.

+---------------------+

| Opening the Page |

+----------+----------+

|

v

+----------+----------+

| Modal with Welcome |

| Message |

+----------+----------+

|

v

+----------+----------+

| Home Page |

+----------+----------+

| | | |

v v v v

+---------+------+----------+---------+

| Profile | Help | Timeline | Article |

+---------+------| Column | Column |

| | | (Left) | (Right) |

v v +---------+----------+

+--------+--------+ | |

| Profile | Help | | |

| Sec. | Sec. | | |

+---------+----- -+ v v

+---------+---------------+

|Timeline | iframe with |

|with | Articles & |

|buttons | Review |

| | Questions |

+---------+---------------+

**Storyboard Pages**

While in development, the overall product design and interface has changed as I addressed challenges in the development. For historic and possible future development, I find it important to see where we started to see what design choices we omitted and design choices that remain intact.

*On ANY screen, the user can select:*

*• \_\_ to minimize*

*• square/middle button to maximize/miniaturize*

*• X to exit the page/browser*

A screenshot of a computer

Description automatically generatedHome

The screen appears as above when entering the webpage.

The page starts at the right most part of the page, or the most recent year on the timeline.

A year on the timeline is filled above the line with an event bubble correlating to that year.

The event titles with be brief but descriptive titles to help identify events immediately.

Event bubbles can be clicked and hovered over for more information and act like navigation links.

The two buttons on the right pane border are shown; The top button is the “profile” that includes cached statistics from the user’s exploration. The bottom button is a help button that explains page by page how to use the website.

There is a text search bar at the bottom of the screen with a “search” button next to it. The user can only type within the text search bar and nowhere else.

The scroll bar at the bottom can move the page left to right.

The user can

* Select “Profile” button.
  + Go to **Profile** page.
* Select “Help” button.
  + Go to **the Help** page.
* Select the text-based search bar.
  + Allow user to input text.
* Select the “Search” button.
  + Go to **Search** page.
* Scroll with scroll bar.
  + Move left or right on the page.
* Scroll with mouse.
  + Move left or right on the page.
* Hover over an event bubble.
* Click an event bubble.

A screenshot of a computer

Description automatically generatedHelp

The screen appears above the **Home** page.

There is a scroll bar to go up and down on the right side of the pop-up page. *NOTE: user cannot scroll right and left anymore or access the* ***Profile*** *page while in the* ***Help*** *page.*

The “Help” button and the “Profile” button are still in display.

Step-by-step guide on how to utilize the website through pictures and text to match the scenario shown.

The user can

* Select “Help” button.
  + Go to previous page or **Home** page.
* Scroll with scroll bar.
  + Move up and down within the **Help** page guide.
* Scroll with mouse.
  + Move up and down within the **Help** page guide.

A screenshot of a computer

Description automatically generatedProfile

The screen appears above the **Home** page with similar functionality.

The “Help” button and the “Profile” button are still in display.

Hide the event boxes behind the user “Profile”.

Show the five user statistics –

Number answered correctly: <show # of correct>

Number answered incorrectly: <show # of incorrect>

Number of problems asked: <show # of problems>

Number of articles explored: <How # of articles read>

Average percent of correct answers: <calculate and show percentage>

The user can

* Select “Profile” button.
  + Closes **Profile** Page
  + Go to the previous page or the **Home** page.
* Select “Help” button.
  + Closes **Profile** Page
  + Go to **the Help** page.
* Scroll with scroll bar.
  + Move left or right on the page.
  + Profile stays on display.
* Scroll with mouse.
  + Move left or right on the page.
  + Profile stays on display.
* Select the text-based search bar.
  + Closes **Profile** Page
  + Allow users to input text.
* Select the “Search” button.
  + Closes **Profile** Page
  + Go to **Search** page.
* Hover over an event bubble.
  + Closes **Profile** Page
  + Go to **Hover Event** Page
* Click an event bubble.
  + Closes **Profile** Page
  + Go to **Event Article** Page

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedSearch

This screen follows the format of the **Home** page. When a search is initialized and it is successful, the user will be displayed a limited view of the timeline. The timeline will display events related to the given user input if the events are available. However, if the event is not available the page remains the same and an error message will be displayed for five seconds showing the following text: “That is unavailable”.

The “Help” button and the “Profile” button are still in display. However, going forward through the pages except the pages directly not claiming them through their description, we will assume they are there within the page set up and with the same functionality as before.

The user can

* Select “Profile” button.
  + Go to the **Profile** Page.
* Select “Help” button.
  + Go to the **Help** page.
* Scroll with scroll bar.
  + Move left or right on the page.
* Scroll with mouse.
  + Move left or right on the page.
* Select the text-based search bar.
  + Allows users to edit current input text.
  + Allow users to input new text.
* Select the “Search” button.
  + Go to **Search** page.
* Hover over an event bubble.
  + Go to **Hover Event** Page
* Click an event bubble.
  + A screenshot of a computer screen

    Description automatically generatedGo to **Event Article** Page

Hover Event

This screen is shown when the user hovers over an event bubble. It is displayed above the **Home** page with similar functionality.

A pop-up screen with a short description on the event is displayed. It may/or may not display a picture. There selected event will turn red from the base color.

The user can

* Select “Profile” button.
  + **Event Hover** page is hidden.
  + Go to the **Profile** Page.
* Select “Help” button.
  + **Event Hover** page is hidden.
  + Go to the **Help** page.
* Scroll with scroll bar.
  + **Event Hover** page is hidden.
  + Move left or right on the page.
* Scroll with mouse.
  + **Event Hover** page is hidden.
  + Move left or right on the page.
* Select the text-based search bar.
  + **Event Hover** page is hidden.
  + Allows users to edit current input text (if any).
  + Allow users to input new text.
* Select the “Search” button.
  + **Event Hover** page is hidden.
  + Go to **Search** page.
* Hover over an event bubble.
  + Current **Event Hover** page is hidden.
  + Displays NEW **Hover Event** Page.
* Click an event bubble.
  + **Current Event Hover** page is hidden.
  + Go to **Event Article** Page

A purple square with text

Description automatically generatedA close-up of a text

Description automatically generatedEvent Article

This screen is a separate screen from the previous pages and is a new tab for the user. It may or may not display a picture. All information is from the related Wikipedia page and follows the same format, but in a different font and overall display of information. When a user’s answers a question after leaving the page, the user will be sent back to this page with a little guidance at the top to help the user answer the question.

The user can

* X to exit the page
  + Go to **the Question** page.
* Scroll with scroll bar.
  + Move left or right on the page.
* Scroll with mouse.
  + Move left or right on the page.
* X on the feedback
  + Feedback Disappears.

A screenshot of a computer

Description automatically generatedQuestion

The screen appears above the **Home** page. All previous functionality is forgone from the **Home** page.

The user is prompted with a question on a pop up bubble related to the previous article. Questions will be fact based asking regards to year, events, and people related to the event.

The two buttons show “Yes” and “No”. However, other questions can have up to four multiple choices.

There is a “go back” button that allows the user to reopen the article without selecting an answer.

The user can

* Select an “Answer” button

If the button selected is correct,

Go to the **Home** page.

If the button selected is incorrect,

Go to **Event Article (Feedback)** screen.

The appropriate counters are updated

* Select “Go back” button

Go to **Event Article** screen.

Under **Sample Session**, you can see the final product in use to visualize the changes I implemented. The changes in the product design include:

* **Interface Redesign:** The overall interface has been updated to provide a different visual and straightforward user-friendly experience. While most basic functionalities remain intact, the design elements, layout, and navigation have been changed.
* **Event Hover:** The article’s are always on display in the right column. Instead of hovering over the timeline event, when you click it, it displays the corresponding article.
* **Navigation:** Instead of a horizontal timeline, it’s vertical to allow more space for the articles.
* **Search Functionality**: Was omitted entirely.
* **Question Prompt**: User no longer needs to answer questions to view the rest of the article/questions. Instead, the questions are at the bottom of the article for ease of review.

# Installation

Discovering History is a web-based application, meaning users can access it directly through their web browser without needing to install any software. To get started, users simply need to visit the website. For developers, the source code is organized into directories for frontend and backend components, which can be set up by cloning the repository and following the setup instructions provided in the documentation.

**For Users:**

**Installation and Setup:**

**Accessing the Website:**

* Users don't need to install anything. They can simply access the website using a web browser.
* They can navigate to the website's URL DiscoveringHistory.com.

**Open a Web Browser:**

* Launch a web browser on your device (e.g., Chrome, Firefox, Safari).

**Enter the URL:**

* Type or paste the website's URL into the address bar of the browser.

**Press Enter:**

* Press Enter or click "Go" to access the website.

**For Developers:**

**Installation and Setup:**

**Code Editor/IDE:**

* Choose a code editor or integrated development environment (IDE) for writing code.
* Download and install the chosen editor/IDE.

**Web Browser:**

* Ensure you have a modern web browser installed for testing and debugging. This can include Chrome, Firefox, Safari, and Edge.

**Getting Access to Source Code:**

**Obtaining Source Code Files:**

* Download the source code files for the website from the GIT repository.
* This includes
  + *Home.html,*
  + *script.js,*
  + *iframe.html,*
  + *iscript.js,*
  + *style.css*

**Organizing Source Code:**

* Extract the downloaded source code files to a directory on your computer.
* Organize the files into the same folder for easy navigation and maintenance.
* (Optional) If files are in separate folders, make sure you adjust the code’s access location.

**Development Products:**

**Development Server (Optional):**

* For local development and testing, consider setting up a local development server using tools like LiveServer in Visual Studio Code to ensure active communication from the *home.html & iframe.html*.

**Setup and Running the Project:**

**Open Source Code in Editor/IDE:**

* Open the source code files in your chosen code editor or IDE.

**Edit and Modify Files:**

* Make any necessary edits or modifications to the HTML, CSS, and JavaScript files to customize the website.

**Save Changes:**

* Save the changes after editing the files.

**Testing and Debugging:**

* Use the web browser to open the HTML files locally and test the changes.
* Use developer tools in the browser for debugging and troubleshooting any issues.

**Deployment (Optional):**

* Once satisfied with the changes, deploy the website to a web server for public access.

# Using the System

Using Discovering History is straightforward and user-friendly. Users are greeted with an interactive timeline on the main page. By clicking on an event on the timeline, users can read detailed articles about historical events and take quizzes to assess their understanding. The statistics page allows users to track their performance over time, while the help section provides additional guidance on navigating and utilizing the platform's features.

**User's Manual:**

**Starting and Accessing the Product:**

**Access the Website:**

1. Open a web browser on your device.
2. Enter DiscoveringHistory.com into the browser's address bar.
3. Press Enter to navigate to the website.

**Using the Product:**

**Navigating the Timeline:**

* On the website's homepage, you'll find a timeline displayed in the left column.
* Each button on the timeline represents a specific year and event in computer history.
* Click on a button to view more information about that event.

**Reading Articles:**

* When you click on a button in the timeline, an article related to that event will be displayed in the right column.
* Read the article to learn more about the historical event and its significance.

**Reviewing Articles:**

* After reading articles, you can review your understanding of the content.
* A small quiz is present below the article, called the Review Quiz.

**Taking the Review Quiz:**

* In the review section, you'll find a series of questions related to the articles you've read.
* Answer each question to the best of your ability.

**Viewing Stats:**

* Click on the "Stats" button to view your statistics.
* Your stats include the number of correct and incorrect answers provided during the review process in the browser’s current session of the website.

**Accessing Help:**

* If you need assistance or mor information about how to use the website, click on the "Help" button.
* The Help modal will provide detailed instructions on how to navigate and utilize the website's features.

# Sample Session

A screenshot of a computer

Description automatically generated1**. Welcome Popup**

Upon visiting the website for the first time, users are greeted with a welcome popup.

A screenshot of a computer

Description automatically generated

**2. Timeline Overview**

After closing the welcome popup, users are taken to the main page. Here, they can see a timeline that highlights significant events. Each event on the timeline corresponds to an article that provides detailed information about the event. Browse through our interactive timeline to explore historical events. Click on an event to read the corresponding article and take quizzes based on the information provided. Upon selecting an event, the user is directed to an article that provides in-depth information about the event.

A screenshot of a computer

Description automatically generated**3. Articles and Quizzes**

Clicking on an event in the timeline brings up an article related to that event. Each article contains detailed information and is followed by a quiz that tests the user’s understanding. After reading the article, the user takes a quiz to test their knowledge. Their performance on the quiz is recorded and displayed on the statistics page, allowing them to see their progress and areas for improvement.

A screenshot of a computer

Description automatically generated**4. Stats Overview**

Users can visit the stats page to see their performance. The stats page displays the number of right and wrong answers for each session, along with overall progress. See how many questions you've answered correctly and how many you've missed. Use this information to improve and challenge yourself further.

A screenshot of a computer

Description automatically generated**5. Help Information**

The help section provides users with information on how to use the timeline and understand their profile statistics and is available to assist users with any questions about the platform's functionalities.

# Modifications

While I initially envisioned adding more features to further enhance the website, various constraints such as my current skill level, time limitations, and, admittedly, a bit of unplanned events have prevented me from fully realizing those plans. However, I'm open to revisiting and expanding upon the website in the future if circumstances allow.

Here are some possible modifications.

1. **Hiding Buttons After Finishing the Quiz:**
   * This modification involves dynamically hiding the buttons related to the quiz after the user completes it. It helps declutter the interface and provides a clear indication to the user that they have finished the quiz. This might further develop into hiding the article as well.

We would use JavaScript to manipulate the visibility of the buttons based on certain conditions. For example, once the user answers the last question and the buttons are identified by the same data variable as the section, the buttons will be hidden with a function. It should contain code like the algorithm below.

// Loop through buttons and hide them

buttonsToHide.forEach(function(button) {

button.style.display = 'none';

});

1. **Adding a Review Button to Display Questions the User Got Wrong:**
   * A new button is added to allow users to review questions they answered incorrectly. It enhances the learning experience by giving users an opportunity to revisit and reinforce their understanding of the content.

Most likely an update would take part in the wrong () function that would unhide a div that was previously hidden when it matches the correct section and question ID.

OR

When comparing the user’s answer index with the correct answer index stored in the question object, it should create a new paragraph element <p> to append the ‘question-container’ to under wrong().

document.getElementById('question-container').appendChild(questionElement);

1. **Randomized Questions:**
   * Randomized questions ensure that the order of questions within each article quiz changes each time the user attempts.

To ensure that each question retains its corresponding answers after shuffling, you can organize your questions and answers as objects within an array with an assigned topic. Each question object would contain properties for the question itself and an array of its corresponding answers, as well as the corresponding section that they belong to, to allow ease of randomization.

One of the first conditions that need to be met is that the array must be manipulating the correct section. Following that condition would be that it isn’t repeating and/or is identified by a unique ID number.

*var quizData = [*

*{*

*section: “topicName”,*

*Qid: 1,*

*question: "Question 1",*

*answers: ["Answer 1", "Answer 2", "Answer 3"],*

*correctAnswerIndex: 3,*

*},];*

1. **Progress Tracking:**
   * Progress tracking features, such as “1/3 questions” provide users with visual feedback on their progress within each article quiz.
2. **Explanations:**
   * Including references back to the article for each quiz question helps users understand why certain answers are correct or incorrect.

To include a reference back to the article for each quiz question, we can add a link or button within the question display area that directs the user back to the corresponding article for review. Most likely with a bit more guidance to show where the answer to the question is. However, it is very similar to how we utilized buttons on the timeline to take the user to the corresponding article like the code block below:

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#eniac')" value="1949 ENIAC">

1. **Timer:**
   * Optionally, for a challenge, adding a timer to the page could encourage the user to complete the reading a quiz faster.

We would have created a new function for the timer, identify what time it is now, allowing the timer to update every second by subtracting the “start” time. If JavaScript does not automatically have a function to turn it into seconds, we may need to manipulate the value to represent seconds. Finally display it in an ever-updating div.

<div id="timer">0:00</div>

1. **Leaderboards:**
   * Introducing leaderboards can further gamify the learning experience and motivate users to improve their quiz scores.

To measure against yourself or others is up to debate, but first I would have to no longer limit my user to the current session and borrow some cookies to calculate the user’s best time. This may include adding a database so scores can be compared to other users. However, that is not in further development. We will be looking at the case it was against the user.

We could have a timer start every time a button was pressed, allowing the page to be “refreshed” to display the questions previously answered. The best time should not be considered until all questions are answered and only replaced if it’s better than the previous best time.

# Troubleshooting

Here are some common problems accessing and using a website, with potential causes and solutions to help troubleshoot your user experience.

| **Symptom** | **Cause** | **Solution** |
| --- | --- | --- |
| Website not loading | Network connection issues | 1. Check your internet connection. |
|  | Website server down | 2. Visit other websites to verify if the issue is specific to this website. |
|  | Browser cache or cookies issues | 3. Clear browser cache and cookies, then try reloading the website. |
|  | Browser outdated or incompatible | 4. Update your browser to the latest version or try using a different browser. |
| Button or link not working | JavaScript disabled | 1. Enable JavaScript in your browser settings. |
|  | Browser compatibility issues | 2. Try using a different browser. |
|  | Website coding errors | 3. Contact the website developer for assistance. |
|  | Incorrect URL | 4. Double-check the URL and try again. |
|  | Server-side issues | 5. Wait for the server to respond or try accessing the website later. |
| Content not displaying correctly | Browser rendering issues | 1. Clear browser cache and reload the page. |
|  | Website design or CSS errors | 2. Refresh the page or try accessing the website from a different device. |
|  | Content blocked by ad blocker | 3. Disable ad blockers or whitelist the website. |
|  | Incompatible browser extensions | 4. Disable browser extensions one by one to identify the culprit. |
|  | Server-side content delivery issues | 5. Contact the website developer for assistance. |
| Slow website performance | Network congestion | 1. Check your internet connection speed using speed testing tools. |
|  | Server overload or high traffic | 2. Try accessing the website during off-peak hours. |
|  | Website hosting or server issues | 3. Contact the website developer for assistance. |
|  | Browser performance issues | 4. Close unnecessary browser tabs or windows, and clear browser cache and cookies. |
|  | Device hardware limitations | 5. Try accessing the website from a different device. |

# References

*Bootstrap Buttons - examples & tutorial*. (n.d.). MDB - Material Design for Bootstrap. Retrieved April 14, 2024, from <https://mdbootstrap.com/docs/standard/components/buttons/#:~:text=You%20can%20use.btn-outline-%2A%20and.btn-rounded%20together%20to%20make%20the>

*How To Create a Timeline*. (n.d.). Www.w3schools.com. Retrieved April 20, 2024, from <https://www.w3schools.com/howto/howto_css_timeline.asp>

jd6699. (2011, July 8). *Override body style for content in an iframe*. Stack Overflow. <https://stackoverflow.com/questions/6494721/override-body-style-for-content-in-an-iframe#:~:text=Basically%2C%20you%20just%20append%20a%20style%20element%20to>

W3Schools. (2019). *How To Make a Modal Box With CSS and JavaScript*. W3schools.com. Retrieved April 14, 2024 from <https://www.w3schools.com/howto/howto_css_modals.asp>

Arcade, A. (2017, May 8). *How to auto increment a number everytime when a button is clicked on other page using javascript*. Stack Overflow. Retrieved March 25, 2024, from <https://stackoverflow.com/questions/43166426/how-to-auto-increment-a-number-everytime-when-a-button-is-clicked-on-other-page>

# Appendix A

<!—HOME.html

Description: Main Page, utilized as a framework for the timeline and articles -->

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Bootstrap Main load-->

<meta name="viewport" content="width=device-width,initial-scale=1" />

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />

<!-- NOT essential? <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" /> -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>

<!-- MY Script JS -->

<script src="script.js"></script>

<!-- MY Style CSS-->

<link rel="stylesheet" type="text/css" href="style.css">

<h1 class ="center">Discovering History</h1>

</head>

<body>

<!-- Start Modal -->

<div id="startMod" class="modal fade" role="dialog">

<div class="modal-dialog">

<!-- Modal content-->

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal">&times;</button>

<h4 class="modal-title"><b>Welcome!</b></h4>

</div>

<!-- Question answers -->

<div class="input-group">

<p>This website is designed as an engaging learning platform that gamifies the exploration of

fascinating technological facts. Our aim is to make learning enjoyable and interactive by

presenting fun and insightful information about various aspects of technology.</p>

<p>Whether you're a tech enthusiast or just curious about the wonders of innovation, this site

offers an immersive experience where you can discover, learn, and be entertained. Join us on

this educational journey as we uncover the exciting world of technology together!</p>

<p> <b><i>NOTE: Most to all source material is derrived from Wikipedia and may be incorrect</i></b>

</p>

</div>

<div class="modal-footer">

<button type="button" class="btn" data-dismiss="modal">Enter</button>

</div>

</div>

</div>

</div>

<!-- Start Modal-->

<button type="button" class="btn" data-toggle="modal" data-target="#profileMod">Stats</button>

<!--<button type="button" class="btn" data-toggle="modal" data-target="#reviewMod">Review</button> -->

<button type="button" class="btn" data-toggle="modal" data-target="#helpMod">Help</button>

<!-- Profile Modal -->

<div id="profileMod" class="modal fade" role="dialog">

<div class="modal-dialog">

<!-- Modal content-->

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal">&times;</button>

<h4 class="modal-title">Profile Stats</h4>

</div>

<!-- modal-body -->

<div class="modal-body">

<h4 for="outputright">Right Answers:</h4>

<p id="outputright"> </p>

<h4 for="outputwrong">Wrong Answers:</h4>

<p id="outputwrong"> </p>

</div>

<div class="modal-footer">

</div>

</div>

</div>

</div>

<!-- Profile Modal-->

<!-- Review Modal -->

<div id="reviewMod" class="modal fade" role="dialog">

<div class="modal-dialog">

<!-- Modal content-->

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal">&times;</button>

<h4 class="modal-title">Review Questions</h4>

</div>

<!-- modal-body -->

<div class="modal-body">

</div>

<div class="modal-footer">

</div>

</div>

</div>

</div>

<!-- Review Modal-->

<!-- Help Modal -->

<div id="helpMod" class="modal fade" role="dialog">

<div class="modal-dialog">

<!-- Modal content-->

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal">&times;</button>

<h4 class="modal-title">Help</h4>

</div>

<!-- modal-body -->

<div class="modal-body">

<p><b>Timeline Column:</b></p>

<p>This column displays a timeline, each correspond to a specific year and event in computer history. When a user clicks on a button, it navigates to its article on the right column. These articles provide a short detailed article about the historical event.</p>

<p><b>Profile:</b></p>

<p>The profile section tracks the user's progress and performance. It records the number of correct and incorrect answers provided during the review process. This feature enables users to monitor their learning progress and identify areas where they may need improvement.</p>

</div>

<div class="modal-footer">

</div>

</div>

</div>

</div>

<!-- Help Modal -->

<p></p>

<div class="row">

<div class="column timeline-column">

<div class="timeline">

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#bug')" value="1947 Bug">

</div>

</div>

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#eniac')"

value="1949 ENIAC">

</div>

</div>

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#harddrive')"

value="1957 Hard Drive">

</div>

</div>

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#spacewar')"

value="1962 Spacewar">

</div>

</div>

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#mouse')"

value="1964 Mouse">

</div>

</div>

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#virus')"

value="1971 Virus">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#email')"

value="1971 Email">

</div>

</div>

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#webcam')"

value="1991 Webcam">

</div>

</div>

<div class="container right">

<div class="content">

<input type="button" class="btn-timeline" onclick="setURL('iframe.html#chess')"

value="1997 Deep Blue">

</div>

</div>

</div>

</div>

<div class="column iframe-column">

<p> <iframe id="iframe" sandbox="allow-scripts allow-same-origin" src="iframe.html" height="500"

width='1800' style="max-width:100%" frameborder='0' marginheight='0' marginwidth='0'></iframe></p>

</div>

</div>

</body>

</html>

<!—iframe.html

Description: Article Page, in correlation to the timeline, you can read articles here -->

<!DOCTYPE html>

<html lang="en">

<head>

<style>

body {

color: #3e4a19;

margin-right: 20px; /\*add space to the right of the iframe\*/

}

article {

background-color: #a98467;

/\* Set your desired background color \*/

padding: 10px;

/\* Add padding for spacing \*/

margin-bottom: 20px;

/\* Add margin for separation between articles \*/

border: 4px solid #6c584c;

border-radius: 60px; /\* rounded corners to the border \*/

}

button {

width: 300px;

height: 50px;

background-color: #adc178;

border: none;

}

.btn-review{

width: 100px;

height: 50px;

background-color: #fdc17f;

border-radius: 100%;

}

.question-container {

color: black;

}

</style>

</head>

<body>

<article>

<section id="virus">

<h3><b>First Computer <i>'Virus'</i></b></h3>

</section>

<p>The first computer virus was called creeper and was created in <i>1971</i>.</p>

<p>Creeper was an experimental computer program written by <b>Bob Thomas</b> in <i>1971</i>. It was designed to move

between mainframe computers system, with a later version by <b>Ray Tomlinson</b> designed to copy itself between

computers rather than simply move.</p>

<p>This self-replicating version of Creeper is generally accepted to be the first computer worm. Creeper was a test

created to demonstrate the possibility of a self-replicating computer program that could spread to other

computers.</p>

<p>The program was not actively malicious software as it caused no damage to data, the only effect being a message

it output to the teletype reading <b>"I'M THE CREEPER : CATCH ME IF YOU CAN"</b></p>

<p>Reaper was the first anti-virus software, designed to delete Creeper by moving across the computers. It was

created by <b>Ray Tomlinson</b> in <i>1972</i>.</p>

</article>

<div class="review" id="virus">

<div class="question-container" id="question1">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op1" data-section="virus">Option 1</button>

<button type="button" class="option next" id="op2" data-section="virus">Option 2</button>

<button type="button" class="option next" id="op3" data-section="virus">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="webcam">

<h3><b>How's my coffee?</b></h3>

<p>The first webcam was used at <b>Cambridge University</b> in <i>1991</i>.</p>

<p>A webcam was pointed at the Trojan Room coffee pot in the <b>Cambridge University</b> Computer Science

Department. Initially operating over a local network instead of the web. </p>

<p>The camera was finally switched off on <i>August 22, 2001</i>. The final image captured by the camera can still

be viewed at its homepage.</p>

</article>

<div class="review" >

<div class="question-container" id="question2">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op4" data-section="webcam">Option 1</button>

<button type="button" class="option next" id="op5" data-section="webcam">Option 2</button>

<button type="button" class="option next" id="op6" data-section="webcam">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="spacewar">

<h3><b>Space War</b></h3>

</section>

<p>The first computer game was called <b>Spacewar!</b> and was created in 1962.

<p><b>Spacewar!</b> is a space combat video game developed in <i>1962</i> by <i>Steve Russell</i> in collaboration

with <i>Martin Graetz, Wayne Wiitanen, Bob Saunders, Steve Piner, and others</i>. It was written for the newly

installed <b>DEC PDP-1</b> minicomputer at the <b>Massachusetts Institute of Technology</b>. After its initial

creation, <b>Spacewar!</b> was expanded further by other students and employees of universities in the area,

including <i>Dan Edwards</i> and <i>Peter Samson</i>. It was also spread to many of the few dozen installations of

the <b>PDP-1</b> computer, making <b>Spacewar!</b> the first known video game to be played at multiple computer

installations. </p>

<p><b>Spacewar!</b> is one of the most important and influential games in the early history of video games. It was

extremely popular in the small programming community in the <i>1960</i>s and the public domain code was widely

ported to and recreated on other computer systems at the time, especially after computer systems with monitors

became more widespread towards the end of the decade. </p>

<p>It directly inspired many other video games, such as the first commercial arcade video games, Galaxy Game and

Computer Space (<i>1971</i>), and later games such as <b>Asteroids</b> (<i>1979</i>). </p>

</article>

<div class="review" >

<div class="question-container" id="question3">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op7" data-section="spacewar">Option 1</button>

<button type="button" class="option next" id="op8" data-section="spacewar">Option 2</button>

<button type="button" class="option next" id="op9" data-section="spacewar">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="mouse">

<h3>The First Mouse</h3>

</section>

<p>The first computer mouse was <b>invented in 1964</b> and was made of wood.</p>

<p>The first public demonstration of a mouse controlling a computer system was done by <b>Doug Engelbart</b> in <b>1968</b> as part of the <i>Mother of All Demos</i>.</p>

<p>Mice originally used two separate wheels to directly track movement across a surface: one in the x-dimension and one in the Y. Later, the standard design shifted to use a ball rolling on a surface to detect motion, in turn connected to internal rollers. Most modern mice use optical movement detection with no moving parts.</p>

</article>

<div class="review" >

<div class="question-container" id="question4">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op10" data-section="mouse">Option 1</button>

<button type="button" class="option next" id="op11" data-section="mouse">Option 2</button>

<button type="button" class="option next" id="op12" data-section="mouse">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="email">

<h3>The First Email</h3>

</section>

<p>The first email was sent in <strong>1971</strong>.</p>

<p>In <strong>1971</strong>, the first ARPANET network mail was sent, introducing the now-familiar address syntax with the '@' symbol designating the user's system address. Over a series of RFCs, conventions were refined for sending mail messages over the File Transfer Protocol. Several other email networks developed in the <strong>1970s</strong> and expanded subsequently.</p>

</article>

<div class="review" >

<div class="question-container" id="question5">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op13" data-section="email">Option 1</button>

<button type="button" class="option next" id="op14" data-section="email">Option 2</button>

<button type="button" class="option next" id="op15" data-section="email">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="eniac">

<h3>30 Ton Computer?!</h3>

</section>

<p>

The first electronic computer weighed more than <b>30 tons</b>.

</p>

<p>

<b>ENIAC</b> was the first programmable, electronic,

general-purpose digital computer, completed in <b>1945</b>.

</p>

<p>

<b>ENIAC</b> was a large, modular computer, composed of individual panels to perform different functions. Twenty of these

modules were accumulators that could not only add and subtract, but hold a ten-digit decimal number in memory.

Numbers were passed between these units across several general-purpose buses (or trays, as they were called). In

order to achieve its high speed, the panels had to send and receive numbers, compute, save the answer and trigger

the next operation, all without any moving parts. Key to its versatility was the ability to branch; it could

trigger different operations, depending on the sign of a computed result.

</p>

<p>

It weighed more than <b>30 short tons (27 t)</b>, was roughly <b>8 ft (2 m)</b> tall, <b>3 ft (1 m)</b> deep, and <b>100 ft (30 m)</b> long,

occupied <b>300 sq ft (28 m2)</b> and consumed <b>150 kW</b> of electricity.

</p>

<p>

<b>ENIAC</b> used common octal-base radio tubes of the day.

</p>

<p>

Several tubes burned out almost every day, leaving <b>ENIAC</b> nonfunctional about half the time. Special

high-reliability tubes were not available until <b>1948</b>. Most of these failures, however, occurred during the warm-up

and cool-down periods, when the tube heaters and cathodes were under the most thermal stress. Engineers reduced

<b>ENIAC's</b> tube failures to the more acceptable rate of one tube every two days. According to an interview in 1989

with <b>Eckert</b>, "We had a tube fail about every two days and we could locate the problem within 15 minutes."In

<b>1954</b>, the longest continuous period of operation without a failure was <b>116 hours</b>—close to five days.

</p>

</article>

<div class="review" >

<div class="question-container" id="question6">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op16" data-section="eniac">Option 1</button>

<button type="button" class="option next" id="op17" data-section="eniac">Option 2</button>

<button type="button" class="option next" id="op18" data-section="eniac">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="harddrive">

<h3> First Harddrive</h3>

</section>

<p>

The first hard disk drive had a capacity of <b>5 megabytes</b> and was the size of a refrigerator.

</p>

<p>

The first production <b>IBM hard disk drive</b>, the <b>350 disk storage</b>, shipped in <b>1957</b> as a component of the <b>IBM 305

RAMAC system</b>. It was approximately the size of two large refrigerators and stored <b>five million six-bit characters

(3.75 megabytes)</b> on a stack of <b>52 disks (100 surfaces used)</b>. The <b>350</b> had a single arm with two read/write

heads, one facing up and the other down, that moved both horizontally between a pair of adjacent platters and

vertically from one pair of platters to a second set.

</p>

<p>

A <b>hard disk drive (HDD)</b>, <b>hard disk</b>, <b>hard drive</b>, or <b>fixed disk</b>, is an electro-mechanical data storage device

that stores and retrieves digital data using magnetic storage with one or more rigid rapidly rotating platters

coated with magnetic material. The platters are paired with magnetic heads, usually arranged on a moving actuator

arm, which read and write data to the platter surfaces. Data is accessed in a random-access manner, meaning

that individual blocks of data can be stored and retrieved in any order. <b>HDDs</b> are a type of non-volatile storage,

retaining stored data when powered off. Modern <b>HDDs</b> are typically in the form of a small rectangular box.

</p>

</article>

<div class="review" >

<div class="question-container" id="question7">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op19" data-section="harddrive">Option 1</button>

<button type="button" class="option next" id="op20" data-section="harddrive">Option 2</button>

<button type="button" class="option next" id="op21" data-section="harddrive">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="chess">

<h3> Beat me in Chess</h3>

</section>

<p>

The first computer to defeat a human in chess was IBM’s Deep Blue in <b>1997</b>.

</p>

<p>

Deep Blue was a chess-playing expert system run on a unique purpose-built IBM supercomputer. It was the first

computer to win a game, and the first to win a match, against a reigning world champion under regular time

controls.

</p>

<p>

Garry Kasparov, <i>Deep Blue</i> played Kasparov

However, Kasparov won three and drew two of the following five games, beating <i>Deep Blue</i> by 4–2 at the close of the

match.

</p>

</article>

<div class="review" >

<div class="question-container" id="question8">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op22" data-section="chess">Option 1</button>

<button type="button" class="option next" id="op23" data-section="chess">Option 2</button>

<button type="button" class="option next" id="op24" data-section="chess">Option 3</button>

</div>

</div>

<p></p>

<article>

<section id="bug">

<h3> A Real?! Bug</h3>

</section>

<p>

The first computer bug was a real insect that got trapped in a computer relay in <b>1947</b>.

</p>

<p>

A software bug is an error, flaw or fault in the design, development, or operation of computer software that

causes it to produce an <i>incorrect</i> or <i>unexpected</i> result, or to behave in <i>unintended</i> ways. The process of finding

and correcting bugs is termed "<i>debugging</i>" and often uses formal techniques or tools to pinpoint bugs. Since the

1950s, some computer systems have been designed to detect or auto-correct various software errors during

operations.

</p>

<p>

The term "<i>bug</i>" was used in an account by computer pioneer Grace Hopper, who publicized the cause of a malfunction

in an early electromechanical computer.

</p>

<p>

Operators traced an error in the Mark II to a moth trapped in a relay, coining the term <i>bug</i>. This bug was

carefully removed and taped to the log book. Stemming from the first bug, today we call errors or glitches in a

program a <i>bug</i>.

</p>

<p>

The operators who found it, including William "Bill" Burke, later of the Naval Weapons Laboratory, Dahlgren,

Virginia, were familiar with the engineering term and amusedly kept the insect with the notation "<i>First actual

case of bug being found.</i>" This log book, complete with attached moth, is part of the collection of the Smithsonian

National Museum of American History.

</p>

</article>

<div class="review" >

<div class="question-container" id="question9">

<h1>Question goes here</h1>

</div>

<div class="option-container">

<button type="button" class="option next" id="op25" data-section="bug">Option 1</button>

<button type="button" class="option next" id="op26" data-section="bug">Option 2</button>

<button type="button" class="option next" id="op27" data-section="bug">Option 3</button>

</div>

</div>

<!-- Script to call functions -->

<script src="iscript.js"></script>

</body>

</html>

<!—script.js

Description: interactive user interface for the main homepage HOME.html

Includes the right/wrong incrementation of statistics page -->

$(document).ready(function () {

$('#startMod').modal('show'); // Show the modal when the document is ready

});

// Define variables to store counts

let rtemp = 0;

let wtemp = 0;

// Function to update the right count in the parent document

function updateValueRight(plusR) {

// Update the right count by adding the provided value

rtemp += plusR;

// Display the updated right count

const rightEl = document.getElementById('outputright');

rightEl.innerText = rtemp;

}

// Function to update the wrong count in the parent document

function updateValueWrong(plusW) {

// Update the wrong count by adding the provided value

wtemp += plusW;

// Display the updated wrong count

const wrongEl = document.getElementById('outputwrong');

wrongEl.innerText = wtemp;

}

/\* set iframe to display certain section of Funfacts.html HAS to be out of the WindDOM\*/

function setURL(url){

console.log("we going here:" + url);

document.getElementById('iframe').src = url;

}

// Initialization code when the DOM is ready

window.addEventListener('DOMContentLoaded', () => {

// Get elements displaying the counts

const rightEl = document.getElementById('outputright');

const wrongEl = document.getElementById('outputwrong');

// Display default counts

rightEl.innerText = rtemp;

wrongEl.innerText = wtemp;

});

<!—iscript.js

Description: interactive user interface for the iframe IFRAME.html

Includes article questions, and communicates with script.js-->

// Function to call the parent function to increment the right count

function right() {

// Call the parent function and pass the incremented value

window.parent.updateValueRight(1); // Increment by 1

}

// Function to call the parent function to increment the wrong count

function wrong() {

// Call the parent function and pass the incremented value

window.parent.updateValueWrong(1); // Increment by 1

}

// Initialization code when the DOM is ready

window.addEventListener('DOMContentLoaded', () => {

const virusQue = [

{

id: 1,

q: "What was the Creeper virus?",

a: ["The Creeper virus was a harmless program designed to improve computer performance by cleaning up unnecessary files.", "Creeper was one of the earliest computer viruses.", "The Creeper virus was a revolutionary virtual reality game."],

correctAnswerIndex: 2 // Second Index

},

{

id: 2,

q: "Who developed Creeper?",

a: ["Creeper was developed by Bob Thomas.", "Creeper's origins remain murky.", "Creeper was thought to have been created by Ray Tomlinson."],

correctAnswerIndex: 1 // First Index

},

{

id: 3,

q: "What was the purpose of the Reaper program?",

a: ["It was actually a secret government surveillance tool disguised as antivirus software.", "It was initially intended to enhance the capabilities of the Creeper virus", "It was designed to detect and remove instances of the Creeper virus from infected systems."],

correctAnswerIndex: 3 // Third Index

},

];

// a:["","",""],

const webcamQue = [

{

id: 1,

q: "What was the webcam's connection to the internet used for?",

a: ["It was part of a research project studying the effects of caffeine consumption.", "To upload images of the coffee pot to the lab's local network, allowing remote viewing via web browsers.", "The webcam's internet connection was utilized to broadcast live cooking shows."],

correctAnswerIndex: 5 // Index of the correct answer

},

{

id: 2,

q: "What was the purpose of the first webcam at Cambridge?",

a: ["To monitor the status of the Trojan Room coffee pot.", "Its purpose was to livestream lectures and seminars.", "The webcam was installed to monitor the weather conditions on campus, providing real-time updates."],

correctAnswerIndex: 4 // Index of the correct answer

},

{

id: 3,

q: "When was the first webcam deployed at the University of Cambridge?",

a: ["The webcam was deployed in 1991, but due to technical issues, it wasn't fully operational until 1992", "The first webcam was deployed in 1990, but its existence wasn't publicly revealed until 1991.", "It was deployed in 1991"],

correctAnswerIndex: 6 // Index of the correct answer

},

];

const spacewarQue = [

{

id: 1,

q: "Who was the primary developer of Spacewar?",

a: ["Steve Russell", "Elon Musk", "Mark Gates"],

correctAnswerIndex: 8 // Index of the correct answer

},

{

id: 2,

q: "How did Spacewar spread to other computer installations?",

a: ["Public Domain Code", "Internet Forms", "Morse Code"],

correctAnswerIndex: 7 // Index of the correct answer

},

{

id: 3,

q: "What is a notable game inspired by Spacewar that was released in 1979?",

a: ["Pac-Man", "Space Invaders", "Asteroids"],

correctAnswerIndex: 9 // Index of the correct answer

},

];

const mouseQue = [

{

id: 1,

q: "What technology do most modern mice utilize for movement detection?",

a: ["Radar", "Optical Sensors", "Sonar"],

correctAnswerIndex: 11 // Second Index

},

{

id: 2,

q: "How did the original mice track movement across a surface?",

a: ["Two separate wheels for x and y dimensions", "Laser that calculates x and y dimensions", "Tiny motors provide vibrational feedback that translated to x and y dimensions"],

correctAnswerIndex: 12 // First Index

},

{

id: 3,

q: "Who invented the first computer mouse, and in what year?",

a: ["Mike Gildon, 1969", "Bill Gates, 1975", "Douglas Engelbart, 1964"],

correctAnswerIndex: 10 // Third Index

},

];

const emailQue = [

{

id: 1,

q: "When was the first email sent, and on which network?",

a: ["The first email was sent in 1975 on Outlook.", "The first email was sent in 1971 on the ARPANET network.", "The first email was sent in 1984 on the internet."],

correctAnswerIndex: 14 // Second Index

},

{

id: 2,

q: "What are some key features of the first ARPANET network mail system?",

a: ["Introduced '@' in email addresses, become a standard of email syntax.", "Integrated with Virtual reality and allowed visual display in VR.", "Capabilities of real-time translation allowed automated translation services."],

correctAnswerIndex: 15 // First Index

},

{

id: 3,

q: "What role did RFCs (Request for Comments) play in the development of email protocols?",

a: ["RFCs serve as a legal framework for email usage, making compatibility an issue in later years.", "RFCs were used as a plateform for advertising, not focusing on technical specifications and protocols.", "RFCs served as a platform for collaboration and consensus-building among researchers and developers, facilitating the establishment of common protocols and standards for email communication."],

correctAnswerIndex: 13 // Third Index

},

];

const eniacQue = [

{

id: 1,

q: "What were some key features of the ENIAC computer?",

a: ["built-in internet connectivity", "general-purpose digital capabilities", "voice recognition capabilities"],

correctAnswerIndex: 17 // Second Index

},

{

id: 2,

q: "What was the longest continuous period of operation without a failure for the ENIAC computer, and when did it occur?",

a: ["64 days in 1950", "116 hours in 1954.", "45 hours in 1946"],

correctAnswerIndex: 18 // First Index

},

{

id: 3,

q: "How much electricity did the ENIAC computer consume?",

a: ["100 kW", "50 kW", "150 kW"],

correctAnswerIndex: 16 // Third Index

},

];

const harddriveQue = [

{

id: 1,

q: "How many read/write heads did the IBM 350 disk storage have?",

a: ["Four read/write heads", "Two read/write heads", "Three read/write heads"],

correctAnswerIndex: 20 // Second Index

},

{

id: 2,

q: "What type of storage is a hard disk drive (HDD) classified as?",

a: ["Non-volatile storage", "Volatile storage", "Temporary storage"],

correctAnswerIndex: 21 // First Index

},

{

id: 3,

q: "How is data accessed on a hard disk drive (HDD)?",

a: ["In a direct-access manner", "In a sequential-access manner", "In a random-access manner"],

correctAnswerIndex: 19 // Third Index

},

];

const chessQue = [

{

id: 1,

q: "Who was the reigning world champion defeated by Deep Blue?",

a: ["Bobby Fischer", "Garry Kasparov", "Anatoly Karpov"],

correctAnswerIndex: 23 // Second Index

},

{

id: 2,

q: "How many games did Kasparov win in the match against Deep Blue?",

a: ["Three games", "Four games", "Two games"],

correctAnswerIndex: 24 // First Index

},

{

id: 3,

q: "What was unique about Deep Blue’s hardware?",

a: ["It was a distributed network of PCs", "It was a standard personal computer", "It was a purpose-built IBM supercomputer"],

correctAnswerIndex: 22 // Third Index

},

];

const bugQue = [

{

id: 1,

q: "Who publicized the cause of the first computer malfunction that was attributed to a 'bug'?",

a: ["Alan Turing", "Grace Hopper", "John von Neumann"],

correctAnswerIndex: 26 // Second Index

},

{

id: 2,

q: "What term is used to describe the process of finding and correcting software bugs?",

a: ["Debugging", "Encoding", "Compiling"],

correctAnswerIndex: 27 // First Index

},

{

id: 3,

q: "What was the first computer bug?",

a: ["A hardware malfunction in a mainframe computer", "A software error in a banking program", "A real insect that got trapped in a computer relay"],

correctAnswerIndex: 25 // Third Index

},

];

const question1 = document.getElementById("question1");

const question2 = document.getElementById("question2");

const question3 = document.getElementById("question3");

const question4 = document.getElementById("question4");

const question5 = document.getElementById("question5");

const question6 = document.getElementById("question6");

const question7 = document.getElementById("question7");

const question8 = document.getElementById("question8");

const question9 = document.getElementById("question9");

let id = 0; // Initialize id here

let id2 = 0; // Initialize id here

let id3 = 0; // Initialize id here

let id4 = 0; // Initialize id here

let id5 = 0; // Initialize id here

let id6 = 0; // Initialize id here

let id7 = 0; // Initialize id here

let id8 = 0; // Initialize id here

let id9 = 0; // Initialize id here

function iterate(id) {

console.log("Inside iterate function. ID:", id);

// Getting the options

const op1 = document.getElementById("op1");

const op2 = document.getElementById("op2");

const op3 = document.getElementById("op3");

// Providing option text

op1.innerText = virusQue[id].a[0]; // Set text of option 1 button

op2.innerText = virusQue[id].a[1]; // Set text of option 2 button

op3.innerText = virusQue[id].a[2]; // Set text of option 3 button

// Setting the question text

question1.innerText = virusQue[id].q;

}

function iterate2(id2) {

console.log("Inside iterate2 function. ID:", id2);

// Getting the options

const op1 = document.getElementById("op4");

const op2 = document.getElementById("op5");

const op3 = document.getElementById("op6");

// Providing option text

op1.innerText = webcamQue[id2].a[0]; // Set text of option 1 button

op2.innerText = webcamQue[id2].a[1]; // Set text of option 2 button

op3.innerText = webcamQue[id2].a[2]; // Set text of option 3 button

// Setting the question text

question2.innerText = webcamQue[id2].q;

}

function iterate3(id3) {

console.log("Inside iterate3 function. ID:", id3);

// Getting the options

const op1 = document.getElementById("op7");

const op2 = document.getElementById("op8");

const op3 = document.getElementById("op9");

// Providing option text

op1.innerText = spacewarQue[id3].a[0]; // Set text of option 1 button

op2.innerText = spacewarQue[id3].a[1]; // Set text of option 2 button

op3.innerText = spacewarQue[id3].a[2]; // Set text of option 3 button

// Setting the question text

question3.innerText = spacewarQue[id3].q;

}

function iterate4(id4) {

console.log("Inside iterate4 function. ID:", id4);

// Getting the options

const op1 = document.getElementById("op10");

const op2 = document.getElementById("op11");

const op3 = document.getElementById("op12");

// Providing option text

op1.innerText = mouseQue[id4].a[0]; // Set text of option 1 button

op2.innerText = mouseQue[id4].a[1]; // Set text of option 2 button

op3.innerText = mouseQue[id4].a[2]; // Set text of option 3 button

// Setting the question text

question4.innerText = mouseQue[id4].q;

}

function iterate5(id5) {

console.log("Inside iterate5 function. ID:", id5);

// Getting the options

const op1 = document.getElementById("op13");

const op2 = document.getElementById("op14");

const op3 = document.getElementById("op15");

// Providing option text

op1.innerText = emailQue[id5].a[0]; // Set text of option 1 button

op2.innerText = emailQue[id5].a[1]; // Set text of option 2 button

op3.innerText = emailQue[id5].a[2]; // Set text of option 3 button

// Setting the question text

question5.innerText = emailQue[id5].q;

}

function iterate6(id6) {

console.log("Inside iterate6 function. ID:", id6);

// Getting the options

const op1 = document.getElementById("op16");

const op2 = document.getElementById("op17");

const op3 = document.getElementById("op18");

// Providing option text

op1.innerText = eniacQue[id6].a[0]; // Set text of option 1 button

op2.innerText = eniacQue[id6].a[1]; // Set text of option 2 button

op3.innerText = eniacQue[id6].a[2]; // Set text of option 3 button

// Setting the question text

question6.innerText = eniacQue[id6].q;

}

function iterate7(id7) {

console.log("Inside iterate7 function. ID:", id7);

// Getting the options

const op1 = document.getElementById("op19");

const op2 = document.getElementById("op20");

const op3 = document.getElementById("op21");

// Providing option text

op1.innerText = harddriveQue[id7].a[0]; // Set text of option 1 button

op2.innerText = harddriveQue[id7].a[1]; // Set text of option 2 button

op3.innerText = harddriveQue[id7].a[2]; // Set text of option 3 button

// Setting the question text

question7.innerText = harddriveQue[id7].q;

}

function iterate8(id8) {

console.log("Inside iterate8 function. ID:", id8);

// Getting the options

const op1 = document.getElementById("op22");

const op2 = document.getElementById("op23");

const op3 = document.getElementById("op24");

// Providing option text

op1.innerText = chessQue[id8].a[0]; // Set text of option 1 button

op2.innerText = chessQue[id8].a[1]; // Set text of option 2 button

op3.innerText = chessQue[id8].a[2]; // Set text of option 3 button

// Setting the question text

question8.innerText = chessQue[id8].q;

}

function iterate9(id9) {

console.log("Inside iterate9 function. ID:", id9);

// Getting the options

const op1 = document.getElementById("op25");

const op2 = document.getElementById("op26");

const op3 = document.getElementById("op27");

// Providing option text

op1.innerText = bugQue[id9].a[0]; // Set text of option 1 button

op2.innerText = bugQue[id9].a[1]; // Set text of option 2 button

op3.innerText = bugQue[id9].a[2]; // Set text of option 3 button

// Setting the question text

question9.innerText = bugQue[id9].q;

}

iterate(id);

iterate2(id2);

iterate3(id3);

iterate4(id4);

iterate5(id5);

iterate6(id6);

iterate7(id7);

iterate8(id8);

iterate9(id9);

// Event listener for all buttons with class 'option.next'

const optionButtons = document.querySelectorAll('.option.next');

optionButtons.forEach(button => {

button.addEventListener('click', () => {

const selectedOptionIndex = parseInt(button.id.slice(-1)); // Extracts the number from the button ID

const section = button.dataset.section; // Get the section from the data-section attribute

let correctAnswerIndex;

console.log("Section:", section);

// Determine the correct array based on the section

if (section === "virus") {

correctAnswerIndex = virusQue[id].correctAnswerIndex; // Get the correct answer index for the current virus question

console.log("virus correct answer is " + correctAnswerIndex);

}

if (section === "webcam") {

correctAnswerIndex = webcamQue[id2].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("webcam correct answer is " + correctAnswerIndex);

}

if (section === "spacewar") {

correctAnswerIndex = spacewarQue[id3].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("spacewar correct answer is " + correctAnswerIndex);

}

if (section === "mouse") {

correctAnswerIndex = mouseQue[id4].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("mouse correct answer is " + correctAnswerIndex);

}

if (section === "email") {

correctAnswerIndex = emailQue[id5].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("email correct answer is " + correctAnswerIndex);

}

if (section === "eniac") {

correctAnswerIndex = eniacQue[id6].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("eniac correct answer is " + correctAnswerIndex);

}

if (section === "harddrive") {

correctAnswerIndex = harddriveQue[id7].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("harddrive correct answer is " + correctAnswerIndex);

}

if (section === "chess") {

correctAnswerIndex = chessQue[id8].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("chess correct answer is " + correctAnswerIndex);

}

if (section === "bug") {

correctAnswerIndex = bugQue[id9].correctAnswerIndex; // Get the correct answer index for the current webcam question

console.log("bug correct answer is " + correctAnswerIndex);

}

console.log("Selected Option Index:", selectedOptionIndex);

console.log("Correct Answer Index:", correctAnswerIndex);

// Right answer!

if (selectedOptionIndex === correctAnswerIndex) {

console.log("you got it");

right();

} else { // Wrong answer!

console.log("you dont got it");

wrong();

}

// Increment to the next question

if (section === "virus" && id < virusQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question virus");

id++; // Increment id

iterate(id); // Update the page with the next virus question

} else if (section === "virus") {

question1.innerText = "Congratulations. You have finished the virus review";

}

if (section === "webcam" && id2 < webcamQue.length - 1) { // Check if it's the webcam section and id2 is within the bounds

console.log("one more question webcam");

id2++; // Increment id2

iterate2(id2); // Update the page with the next webcam question

} else if (section === "webcam") {

question2.innerText = "Congratulations. You have finished the webcam review";

}

// Increment to the next question

if (section === "spacewar" && id3 < spacewarQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question spacewar");

id3++; // Increment id

iterate3(id3); // Update the page with the next virus question

} else if (section === "spacewar") {

question3.innerText = "Congratulations. You have finished the spacewar review";

}

// Increment to the next question

if (section === "mouse" && id4 < mouseQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question mouse");

id4++; // Increment id

iterate4(id4); // Update the page with the next virus question

} else if (section === "mouse") {

question4.innerText = "Congratulations. You have finished the mouse review";

}

// Increment to the next question

if (section === "email" && id5 < emailQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question email");

id5++; // Increment id

iterate5(id5); // Update the page with the next virus question

} else if (section === "email") {

question5.innerText = "Congratulations. You have finished the email review";

}

// Increment to the next question

if (section === "eniac" && id6 < eniacQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question eniac");

id6++; // Increment id

iterate6(id6); // Update the page with the next virus question

} else if (section === "eniac") {

question6.innerText = "Congratulations. You have finished the eniac review";

}

// Increment to the next question

if (section === "harddrive" && id7 < harddriveQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question harddrive");

id7++; // Increment id

iterate7(id7); // Update the page with the next virus question

} else if (section === "harddrive") {

question7.innerText = "Congratulations. You have finished the harddrive review";

}

// Increment to the next question

if (section === "chess" && id8 < chessQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question chess");

id8++; // Increment id

iterate8(id8); // Update the page with the next virus question

} else if (section === "chess") {

question8.innerText = "Congratulations. You have finished the chess review";

}

// Increment to the next question

if (section === "bug" && id9 < bugQue.length - 1) { // Check if it's the virus section and id is within the bounds

console.log("one more question bug");

id9++; // Increment id

iterate9(id9); // Update the page with the next virus question

} else if (section === "bug") {

question9.innerText = "Congratulations. You have finished the bug review";

}

});

});

});

<!—style.css

Description: How elements are to be displayed on the website, main focus on timeline effect -->

/\*content of modals don't touch borders\*/

.input-group {

padding-top: 10px;

padding-bottom: 10px;

padding-right: 10px;

padding-left: 10px;

}

.btn{

color: #3e4a19;

background-color: #b1ce61;

}

/\* timeline buttons circular \*/

.btn-timeline {

color: #3e4a19;

background-color: #b1ce61;

height: 25px;

width: 125px;

border: 1px;

border-radius: 10px;

}

h1.center{

text-align: center;

color: #343f14

}

/\* timeline on left | iframe on right\*/

.row {

display: flex;

}

.column {

flex: 1; /\* This will make both columns equal width \*/

}

.timeline-column {

flex: 1; /\* Adjust the flex value to control the width of the timeline column \*/

}

.iframe-column {

flex: 3; /\* Adjust the flex value to control the width of the iframe column \*/

}

body {

box-sizing: border-box;

background-color: #a4795b;

color: #343f14;

font-family: Helvetica, sans-serif;

}

/\* The location of the line that represents the timeline \*/

.timeline {

position: relative;

max-width: 1200px;

margin: 0 auto;

}

/\* The actual timeline (the vertical ruler) \*/

.timeline::after {

content: '';

position: absolute;

width: 6px;

background-color: #b1ce61;

top: 0;

bottom: 0;

left: 25px;

margin-left: -3px;

}

/\* Container around content \*/

.container {

padding: 10px 10px;

position: relative;

width: 150px;

right: 18%;

background-color: #a4795b;

}

/\* Add arrows to the right container (pointing left) \*/

.right::before {

content: " ";

height: 0;

position: absolute;

top: 13px;

width: 0;

z-index: 1;

left: 4px;

border: medium solid #b1ce61;

border-width: 10px 10px 10px 0;

border-color: transparent #b1ce61 transparent transparent;

}

/\* Media queries - Responsive timeline on screens less than 600px wide \*/

@media screen and (max-width: 600px) {

/\* timeline on left | iframe on right\*/

.row {

display: flex;

}

.column {

flex: 1; /\* This will make both columns equal width \*/

}

.timeline-column {

flex: 1; /\* Adjust the flex value to control the width of the timeline column \*/

}

.iframe-column {

flex: 2; /\* Adjust the flex value to control the width of the iframe column \*/

}

/\* Container around content \*/

.container {

padding: 10px 10px;

position: relative;

width: 150px;

right: 0%;

background-color: #a4795b;

}

}