1/14/25

- Fitt's law- time = *a* + *b* * log(*D* / *S* + 1)
  - D = distance and S = size
  - easier to move mouse to something that is closer and bigger. log(D/S)
- Steering law- time = *a* + *b* * *D* / *W*
  - D = distance (length of tunnel) and W = width (width of tunnel)
  - harder to navigate dropdown menu that is long and wide
- Recognition is faster than recall

- Gulf of execution- discrepancy between what a user wants to do and what they can actually do
- Gulf of evaluation- discrepancy between what the user knows/understands about the state of the system and the actual state of the system

- Affordance- what a system lets you do
- Signifier- signals where and how interaction should take place
- Metaphor- signifier that takes advantage of users' existing knowledge
- Learned association- signifier that user knows from previous experience

- Learnability- Can you get better at using the interface?
- Efficiency- Can you get the job done quickly + well?
- Discoverability- Can you find new/existing UI features/tools?
- Understandability - Do you understand what is happening?

11/16/25

- Selectors can be a little more complicated
  - p, #helloWorld selects all <p> tags and the element with ID "helloWorld"
  - p #helloWorld selects an element with ID "helloWorld" that's inside a <p> tag
  - p#helloWorld selects all <p> tags with ID "helloWorld"
- Position property values
  - static (default)
  - relative- position element relative to its normal/original position
    - top: 10px moves the element 10 pixels down from its original position
  - fixed- element stays in the same position even if the page scrolls
    - right: 0 positions the element 0 pixels away from the right edge of the screen
  - sticky (probably won't use this one often)
  - absolute- position element relative to its nearest ancestor that has a specified position.
    - "Specified position" means you explicitly wrote out the element's position property and assigned it to something besides static.
    - Common use: assign position: relative to the parent element, then assign position: absolute to the child element. Then you can position the child element inside the parent element much like you can position an element with a fixed position inside a webpage.
- With flexboxes (edit position of elements that are children of this container):
  - Choose how items are positioned horizontally using the "justify-content" property
  - Choose how items are positioned vertically using the "align-items" property
  - Choose whether items should be in a row or a column with the "flex-direction" property (default is row, column reverses justify-content and align-items)
    - Specific children elements can override this using the "align-self" property
  - Choose whether items wrap using the "wrap" property (whether they all go on one line or multiple lines)
  - Choose how much space is between lines using the "align-content" property

1/23/25

- jQuery
  - $('h1') selects all h1 tag elements
  - $('#item') selects the element with ID 'item'
  - $('.items') selects all elements with class 'items'
  - $('button') selects all elements with type 'button'
  - $('[href]') selects all elements with an href attribute
  - $('[value="Submit"]') selects all elements whose value attribute has value "Submit"
  - $(document) selects whole document
  - $('#item').html() gets the element's text
  - $('#item').click(callbackFunc) causes callbackFunc to be called every time the element is clicked
  - $('#item').css("property", "value") sets the element's property to the specified value
  - $('#item').append(element)

1/28/25

- One browser process per window (think: everything above and including the bookmarks bar), one renderer process per tab (think: everything below the bookmark bar)
- Browser process knows user inputs. Once it detects an event, it tells the renderer process that the event occurred, and then the renderer process handles that event.
- AJAX allows you to update parts of a webpage without reloading the whole page.
- There is an event loop that executes code. First it runs all the synchronous code, then it runs all the code in the microtask queue (promises), then it runs all the code in the macrotask queue (everything else, like setTimeout, setInterval, ajax). Before code/functions enter either of these queues, we first wait on the web API to return a response.
  - For example, while we're waiting for a promise to resolve, that promise is just waiting on the web API. But once it's resolved, if there is a callback function that is called once that promise resolves, it would go in the microtask queue.

1/30/25

- Four stages of Human-centered design- need finding, iteration, prototyping, testing
- Can try to understand users through observation, surveys, or interviews (either structured, semi-structured, or unstructured).

2/4/25

- Ask targeted, specific questions- easier to answer than general questions
- Affinity diagrams:
  - 1. Capture the main idea of each interview. Your (team's) interpretation
    - Each interpretation note should contain one idea, be self-contained, relate to the participant's motivation, and use first-person pronouns (e.g "I").
  - 2. Group interviews that are similar to each other into clusters called affinity groups
  - 3. Label each affinity group
    - Group hierarchichally; data (white) > category (blue) > theme (pink) > insight (green)
      - 3-7 white notes per blue note
      - Pink and green labels should be about an issue or area of concern
    - Make labels based on the data. Don't try to make categories that fit your pre-conceived labels.

2/6/25

- IDEO rules:
  - 1. Defer judgement
  - 2. Encourage wild ideas
  - 3. Build on others' ideas
  - 4. Stay focused on topic
  - 5. Conversations take place one at a time
  - 6. Be visual
  - 7. Go for quantity
- Speed dating- discover what users like and dislike
  - Done by showing users storyboards
    - 3-4 panels, being handrawn is fine
    - Include some riskier/less rational designs
    - Storyboards do not need to focus on *how* the technology works, but more so on how it's used/user experience
    - Ask them what they like/dislike about the proposed solution; do not ask them about how they would want to improve it. Do not consider it an evaluation of the proposed design.
      - Bad questions: Do you think an Uber-like ride-sharing system would be helpful to UM students who find themselves in such a situation? [Show a picture of the proposed solution with a detailed screen design]. Which features described here would you have the most trouble using?
      - Good questions: Have you ever found yourself nervous about a safe way to get home from north campus in the evening? [Show a picture of the proposed storyboard/solution]. Which aspects of the solution do you find compelling?
  - Goals: need validation and getting user's thoughts on proposed solutions

2/25/25

- Discount usability methods- not user testing, but fast and cheap
  - Cognitive walkthroughs
  - Heuristic evaluations
- Nielson's 10 usability heuristics
  - Visibility of system status
    - The user should know what's going on inside the system
  - Match between system and real world
    - The design should speak the users' language
  - User control and freedom
    - Should have a way to easily undo a user mistake
  - Consistency and standards
    - Users shouldn't have to worry whether different words/situations/actions mean the same thing
  - Error prevention
    - Prevent the user from making a mistake (ex: github making you confirm when you want to delta a repo, autocorrect)
  - Recognition rather than recall
    - List all options out rather than making the user remember them (ex: autocomplete)
  - Flexibility and efficiency of use
    - Shortcuts- hidden from novice users- can speed up interactions
  - Aesthetic and minimalist design
    - Should not include irrelevant/rarely needed information
  - Help users recognize, diagnose, and recover from errors
    - Error messages should be in plain language, precisely indicate problem, and suggest a solution
  - Help and documentation
    - If a user needs this, something is wrong with the interface. But if they need it, give them good help.

3/11/25

- Think-aloud protocol- ask the user to think out loud while using a prototype of your user interface
  - The task you ask them to do should be common and concrete
  - Before starting the think-aloud, introduce your app and explain how to do the think-aloud
    - Tell them to try to not explain things, just verbalize their thoughts
    - Tell them I won't help unless absolutely necessary
    - Describe the task- give written instructions
  - Don't interrupt them while they do this- only ask questions about why did they did certain things after the session is over
  - Remind them to say their thoughts out loud if they haven't spoken in a while
  - Take notes and look for users' strategies and mental models
- Usability finding templates
  - Make each time you view behavior/see evidence of positive or problematic experience
    - Give it unique UFT ID, name, and connect them at the end
    - Make note of frequency (common?), impact (easy to overcome?), persistence (one-time problem?)

3/13/25

- Vue.js - a JavaScript framework for making webpages. Simpler than React.js
- Basic setup steps:
  - 1. Create app
  - 2. Create data
  - 3. Create methods
  - 4. Connect the Vue instance to an HTML element
- Can put JavaScript expressions in {{}}, such as keys in the dictionary that data() returns
- Directives coordinate program flow
  - v-bind
    - Allows you to access Vue data in HTML element attributes (if a string in the attribute value is one of the data keys, it will be replaced with the corresponding value)
    - Can use v-bind: or just :
    - Can bind a class
      - :class = "{disabledButton: inventory == 0, lowStock: inventory > 5}"
        - disabledButton and lowStock are class names
  - v-if
  - v-for
  - v-on

3/18/25

- Bootstrap is responsive- looks good even if you change screen size
- Organizes page into 12 columns
- Classes include:
  - container (margin)
  - container-fluid (no margin)
  - bg-primary, bg-success, bg-danger (set background color)
  - mt-4 (margin top)
  - Text-white
  - p-3 (padding size)
  - nav-X (navigation)
  - btn btn-X (buttons)
  - dropdown-menu
  - dropdown-item
- JSON- string that represents an object that follows strict syntax
- JSON.parse and JSON.stringify
- In JSON, strings need to be in double quotes (in JS, they can be in single quotes)
- JSON cannot contain functions
- If your JavaScript object references another object, JSON includes the referenced object
-
- Get around this by adding ID property, then compare IDs

3/20/25

- Model-View-Controller
  - Model- system state (ex: JSON, SQL database)
  - View- what users see (ex: spreadsheet view, pie chart)
  - Controller- how stuff gets changed (detects input/events, updates stuff accordingly)
  -
- View should be separated from domain logic and model so that the same view can be used across many applications
- View and model should be decoupled
- There are variants of MVC such as MVVM, but they all have separation of concerns, modularity, and reusability
- Tool- software program/utility that helps solve a problem.
  - Ex: VS Code
  - Usually language-agnostic.
- Library- collection of functions that help solve a problem.
  - Ex: jQuery
  - Usually language-specific
- Framework- scaffolds the creation of a tool or application. Has initial components and definable slots that developers can use to accomplish a task
  - Ex: Vue, Bootstrap, React
  - Usually language-specific
- Inversion of control. Library- you get more control, Framework- it gets more control

3/25/25

- Lecture reviews design and development process of a user interface that uses AI to assist feedback provision
- AI gives ECON 101 graders potential feedback they can give to students for essays they are grading
- Users (graders) can see each rubric item on screen. Highlighted green if AI thinks student met that criteria, otherwise red, and additional feedback.
- 4 types of changes made
  - Changes of UI elements
    - Minimize cursor travel
    - Reduce unnecessary buttons (make existing UI elements interactive)
    - Visualizations for discoverability and understandability
  - Changes based on users' mental models
  - Changes for responsible use of AI
  - Changes based on technical feasibilities

3/27/25

- About 1 billion people (15%) have a disability
- Disability- mismatched interaction between someone and their context
- Accessibility- tools that help people navigate mismatched interactions
- Inclusive design- framework that helps design more accessible products
- Assistive technology- improve ability of disabled people
  - Screen readers, screen adjustment, speech input, keyboarding, etc.
- Universal design- one design works for everybody. Seven principles
  - Equitable use
  - Flexibility in use
  - Simple and intuitive use
  - Perceptible information
  - Tolerance and error
  - Low physical effort
  - Size and space for approach and use
- Ability-based design- technology that adapts based on ability of user
- Accessible design- people with disabilities explicitly considered when designing

4/1/25

- AI has uncertain and unpredictable output
- AI lack common sense/abstract concepts to refine interpretation
- Machine learning is dynamic and solutions are not always motivated by real problems
- UI can try to correct for mistakes in AI output, but training an AI to minimize error would be better
- Guidelines for AI-Human interaction
  - Initially
  - During interaction
  - When wrong
  - Over time
- With LLMs, can have narrow gulf of execution but wide gulf of evaluation
- Gulf of envisioning- gulf between what a user envisions/wants and their ability to express it
- Design patterns
  - Visually track prompts and outputs
  - Suggest ideas for prompting
  - Provide multiple outputs
  - Make the output explainable
  - Use domain-specific prompting strategies
  - Allow manual control of output

4/3/25

- Social computing has:
  - Roles
  - Incentives and rewards
  - Norms (informal rules that govern behavior)
    - May need moderation to enforce formal rules
  - Social regulation (how do you get users and keep them?)
    - Social facilitation- to solve cold start problem, make it seem like there's others/it's active and you'll want to join too
    - Want to be able to scale to lots of users

Q10. _____ When developing user interfaces, we need to consider human capabilities and limits. Which of the following is NOT correct about human capabilities that may influence UI design?

- One's working memory can only hold a certain amount of information so that UIs should not create overloads to users' working memory.
- Recognition is easier than recall so that UIs should avoid having users recall a lot of information.
- It takes a certain amount of time for a user to process visual inputs so that visual elements in UIs shouldn't be updated too quickly.
- Users process visual information more efficiently than auditory information so that it's better to present users with images plus captions than images with narration.

Which of the following is correct about why we want to use front-end frameworks such as Vue in web development?

- It helps speed up the rendering of web pages.
- It helps reduce the amount of explicit coordination between model, view, and controller.
- It helps create a one-to-one mapping between model and view, so that it is more friendly for novice programmers of UI.
- It helps separate the developing efforts on model, view and controller, but it does not reduce the coordination between them.

```
<body>
  <div id="app" class="d-flex flex-row">
    <!-- Side Bar -->
    <div class="d-flex flex-column flex-shrink-0 p-3 bg-body-tertiary side-bar
sticky-top">
      <!-- Title -->
      <div class="d-flex align-items-center mb-3 mb-md-0 me-md-auto">
        <span class="fs-2 fw-bold">Artist Search 493</span>
      </div>
      <hr>
      <!-- Search Bar -->
      <div class="input-group mb-3" id="search-bar">
        <div class="input-group-prepend">
          <span class="input-group-text px-2">&#128269;</span>
        </div>
        <div class="dropdown w-84">
          <!-- for how v-model works, see https://v2.vuejs.org/v2/guide/forms -->
          <input type="text" class="form-control px-2 dropdown-toggle"
data-bs-toggle="dropdown" aria-expanded="false" placeholder="type artist name and
enter" v-model="userSearchQuery" @keyup="checkKeyPressed"/>
          <ul class="dropdown-menu">
            <li v-for="(item,idx) in selectedSearchHistory">
              <a class="dropdown-item" @mouseenter="setHoveredButton(item)"
@mouseleave="setHoveredButton(null)">
                <div class="row">
                  <div class="col-sm-10" @click="search(item)">{{ item }}</div>
                  <div v-show="isHoveredButton(item)" class="col-sm-2"
@click="removeFromSearchHistory(item)"> x </div>
                </div>
              </a>
            </li>
          </ul>
        </div>
      </div>

      <!-- Favorites -->
      <div id="favorites" class="mb-1">
        <button class="btn btn-toggle d-inline-flex align-items-center rounded border-0
collapsed gap-1 m-2"
          data-bs-toggle="collapse" data-bs-target="#favorite-collapse"
aria-expanded="true">
          <img src="./assets/Library.svg" />
          Favorites
        </button>
        <div class="collapse show" id="favorite-collapse">
          <ul class="btn-toggle-nav list-unstyled fw-normal pb-1">
            <li v-for="(track,idx) in favoritedTracks">
              <div class="nav" @mouseenter="setHoveredButton(track.trackId)"
@mouseleave="setHoveredButton(null)">
                <div @click="togglePlay(track)">
                  <img v-if="isHoveredButton(track.trackId) && isPlaying(track)"
class="btn play-btn" src="./assets/Pause.svg" alt="pause"/>
                  <img v-else-if="isHoveredButton(track.trackId) && !isPlaying(track)"
class="btn play-btn" src="./assets/Play.svg" alt="play"/>
                  <img v-else class="play-btn" :src="track.artworkUrl100" />
                </div>

                <div class="one-favorite">
                  <div :class="{'scrolling-text': isHoveredButton(track.trackId) ||
isPlaying(track)}">
                    <strong>{{ getObjectProperty(track, "trackName") }}</strong>
                    <div>{{ getObjectProperty(track, "artistName") }}</div>
                  </div>
                </div>

                <div v-show="isHoveredButton(track.trackId)"
@click="toggleFavorited(track)"> x </div>

              </div>
            </li>
          </ul>
        </div>
      </div>
    </div>

    <!-- Main Page -->
    <div class="flex-fill">
      <!-- Top Bar -->
      <nav class="navbar navbar-default sticky-top">
        <div class="container">
          <!-- Music Controls -->
          <div id="music-controls">
            <button disabled>
              <img src="./assets/Shuffle.svg" alt="shuffle" width="15px" height="15px"
/>
            </button>
```

```
            <button disabled>
              <img src="./assets/Back.svg" alt="back" />
            </button>
            <button
              <img v-if="isPlaying(currentlyPlayingTrack)" src='./assets/Pause.svg'
alt="pause" @click="togglePlay(currentlyPlayingTrack)"/>
              <img v-else src='./assets/Play.svg' alt="play"
@click="togglePlay(currentlyPlayingTrack)"/>
            </button>
            <button @click="playNextTrack()">
              <img src="./assets/Fwd.svg" alt="fwd" />
            </button>
            <button @click="toggleRepeat()">
              <img v-if="repeating" src='./assets/Repeating.svg' alt="repeating"
width="15px" height="15px" />
              <img v-else src='./assets/Repeat.svg' alt="repeat" width="15px"
height="15px" />
            </button>
          </div>

          <!-- Currently Playing Track -->
          <div class="p-3 m-1 flex-grow-1 ">
            <img v-if="currentlyPlayingTrack == null" src="./assets/Song.svg" />
            <div v-else class="nav justify-content-center">
              <div>
                <img class="play-btn" :src="currentlyPlayingTrack.artworkUrl100" />
              </div>
              <div class="text-center">
                <strong>{{ getObjectProperty(currentlyPlayingTrack, "trackName")
}}</strong>
                <div>{{ getObjectProperty(currentlyPlayingTrack, "artistName") }} - {{
getObjectProperty(currentlyPlayingTrack, "collectionName") }}</div>
              </div>
            </div>

            <div class="progress" :style="{ width: progress + '%' }"></div>
          </div>

          <!-- Volume Controls -->
          <div id="sound" class="d-flex flex-row align-items-center gap-1">
            <img src="./assets/VolumeDown.svg" alt="volume-down" width="15px"
height="15px" />
            <input type="range" min="1" max="100" v-model="volume"
@input="updateVolume">
            <img src="./assets/VolumeUp.svg" alt="volume-up" width="15px" height="15px"
/>
          </div>
        </div>
      </nav>

      <!-- Artist Search Results -->
      <div class="container">
        <div class="row">

          <!-- Genre Buttons -->
          <div class="col-lg-12 col-md-12 col-sm-12">
            <h4>
              Search Result (Total {{ processedResults.length }} Found.)
            </h4>
            <div>
              <button class="btn btn-outline-success" :class="{active:
selectedGenres.has('ALL')}" @click="toggleGenre('ALL')">ALL</button>
              <button v-for="(genre,idx) in searchResultsGenres" class="btn
btn-outline-primary" :class="{active: selectedGenres.has(genre)}"
@click="toggleGenre(genre)"> {{ genre }} </button>
            </div>
            <h4>
              List of results
            </h4>

            <div class="dropdown">
              <button class="btn btn-info dropdown-toggle" type="button"
data-bs-toggle="dropdown" aria-expanded="false">
                Sort list of results by:
              </button>
              <ul class="dropdown-menu">
                <li><a class="dropdown-item" :class="{active: sortBy == 'Reset to
original'}" @click="sortSearchResults('Reset to original')">Reset to original</a></li>
                <li><a class="dropdown-item" :class="{active: sortBy == 'Collection'}"
@click="sortSearchResults('Collection')">Collection</a></li>
                <li><a class="dropdown-item" :class="{active: sortBy == 'Price'}"
@click="sortSearchResults('Price')">Price</a></li>
              </ul>
            </div>
          </div>

          <!-- Tracks -->
          <div v-for="(item,index) in processedResults" :key="item.trackId"
class='col-lg-6 col-md-6 col-sm-6 artist'>
            <div class="row">

              <div class='artist-albumcover col-sm-4'>
                <img :src="item.artworkUrl100">
              </div>

              <div class='artist-info col-sm-6'>
                <ul class="nav nav-tabs">
                  <li class="nav-item">
                    <button class="nav-link active" data-bs-toggle="tab" type="button"
:href="concatenateStrings('#description-', item.trackId)">Description</button>
                  </li>
                  <li class="nav-item">
                    <button class="nav-link" data-bs-toggle="tab" type="button"
:href="concatenateStrings('#trackInfo-', item.trackId)">Track info</button>
                  </li>
                </ul>

                <div class="tab-content">
                  <div :id="concatenateStrings('description-', item.trackId)"
class="tab-pane active">
                    <div><strong>Artist:</strong> {{ getObjectProperty(item,
"artistName") }}</div>
                    <div><strong>Track:</strong> {{ getObjectProperty(item,
"trackName") }}</div>
                    <div><strong>Collection:</strong> {{ getObjectProperty(item,
"collectionName") }}</div>
                    <div><strong>Price:</strong> {{ getObjectProperty(item,
"trackPrice") }}</div>

                    <ul class="nav">
                      <li class="nav-item">
                        <button class="btn btn-outline-secondary" :class="{active:
isPlaying(item)}" @click="togglePlay(item)">
                          <img v-if="isPlaying(item)" src="./assets/Pause.svg"
alt="pause" class="w-84">
                          <img v-else src="./assets/Play.svg" alt="play" class="w-84">
                        </button>
                      </li>
                      <li class="nav-item">
                        <button class="btn btn-outline-secondary" :class="{active:
isFavorited(item)}" @click="toggleFavorited(item)">
                          <div v-if="isFavorited(item)" class="w-84"> ★ </div>
                          <div v-else class="w-84"> ☆ </div>
                        </button>
                      </li>
                    </ul>
                  </div>

                  <div :id="concatenateStrings('trackInfo-', item.trackId)"
class="tab-pane">
                    <div><strong>Track ID:</strong> {{ getObjectProperty(item,
"trackId") }}</div>
                    <div><strong>Country:</strong> {{ getObjectProperty(item,
"country") }}</div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
    <script src="script.js"></script>
</body>

</html>

const { createApp } = Vue

createApp({

  data() {
    return {
      userSearchQuery: "",
      searchResults: [],
      processedResults: [],
      searchResultsGenres: new Set(),
      selectedGenres: new Set(["ALL"]),
      sortBy: "Reset to original",
      searchHistory: [],
      selectedSearchHistory: [],
      hoveredButton: null,
      favoritedTracks: [],
      currentlyPlayingTrack: null,
      currentlyPlayingTrackAudio: null,
      repeating: false,
      volume: 50,
      progress: 0
    }
  },

  methods: {
    checkKeyPressed: function(event) {
      if (event.keyCode == 13) {
        this.search(this.userSearchQuery);
        this.searchHistory = this.searchHistory.filter(el => el !=
this.userSearchQuery);
        this.searchHistory.unshift(this.userSearchQuery);
        this.selectedSearchHistory = this.searchHistory.slice(0, 5);
      }
    },

    search: function(query) {
      this.userSearchQuery = query;
      URLEncodedQuery = query.replaceAll(" ", "+")

fetch(`https://itunes.apple.com/search?term=${URLEncodedQuery}&media=music&attribute=ar
tistTerm&origin=*`)
        .then(res => res.json())
        .then(resJSON => {
          console.log(resJSON);
          this.searchResults = resJSON.results;
          if (resJSON.resultCount == 0) {
            alert("No artists were found with the keyword.");
          }
          this.searchResultsGenres.clear();
          this.selectedGenres.clear();
          this.selectedGenres.add("ALL");
          for (const searchResult of this.searchResults) {
            this.searchResultsGenres.add(searchResult.primaryGenreName);
          }
          this.sortBy = "Reset to original";
          this.processedResults = [...this.searchResults];
        })
        .catch(err => {
          this.searchResults = [];
          this.searchResultsGenres.clear();
          this.selectedGenres.clear();
        })
    },

    sortSearchResults: function(sortBy) {
      this.sortBy = sortBy;
      switch (sortBy) {
        case 'Reset to original':
          this.filterSearchResults();
          break;
        case 'Collection':
          this.processedResults = this.processedResults.sort((item1, item2) =>
item1.collectionName.localeCompare(item2.collectionName));
          break;
        case 'Price':
          this.processedResults = this.processedResults.sort((item1, item2) =>
item1.trackPrice - item2.trackPrice);
          break;
        default:
          // alert(`You cannot sort by ${sortBy}`);
          break;
      }
    },

    removeFromSearchHistory: function(item) {
      this.searchHistory = this.searchHistory.filter(el => el != item);
      this.selectedSearchHistory = this.searchHistory.slice(0, 5);
    },

    isFavorited: function(track) {
      return this.favoritedTracks.findIndex(el => el.trackId == track.trackId) != -1;
    },

    togglePlay: function(track) {
      if (track == null) {
        return;
      }

      if (this.currentlyPlayingTrackAudio != null &&
this.currentlyPlayingTrackAudio.src == track.previewUrl) {
        this.currentlyPlayingTrackAudio.paused ? this.currentlyPlayingTrackAudio.play()
: this.currentlyPlayingTrackAudio.pause();
        // while .play() and .pause() change the .paused attribute, causing isPlaying()
to return a different value, Vue doesn't recognize that those methods will cause a
change in tracked state because i'm not directly modifying the value of any tracked
variables
        this.$forceUpdate()
        return;
      }
      if (this.currentlyPlayingTrackAudio != null) {
        this.currentlyPlayingTrackAudio.pause();
        this.currentlyPlayingTrackAudio = null;
      }
      this.currentlyPlayingTrackAudio = new Audio(track.previewUrl);
      this.currentlyPlayingTrack = track;
      this.currentlyPlayingTrackAudio.volume = this.volume / 100;
      this.currentlyPlayingTrackAudio.ontimeupdate = () => {
        this.progress = this.currentlyPlayingTrackAudio.currentTime /
this.currentlyPlayingTrackAudio.duration * 100
      }
      this.currentlyPlayingTrackAudio.onended = () => {
        if (this.repeating) {
          this.currentlyPlayingTrackAudio.play();
          return;
        }
        this.currentlyPlayingTrackAudio = null;
        this.$forceUpdate()
      }
      this.currentlyPlayingTrackAudio.play();
    },

    playNextTrack: function() {
      if (this.searchResults.length < 2) {
        return;
      }

      let randomIdx = Math.floor(Math.random() * this.searchResults.length);
      this.togglePlay(this.searchResults[randomIdx]);
    },

    updateVolume: function() {
      if (this.currentlyPlayingTrack == null) {
        return;
      }
      this.currentlyPlayingTrackAudio.volume = this.volume / 100;
    }

  }
}).mount('#app')

// some P3 code
$(document).ready(function () {
  // jQuery selectors
  game_window = $('.game-window');
  game_screen = $("#actual-game");
  asteroid_section = $('.asteroidSection');
});

$("html").on("click", ".restart-button", function(event){
  pausePage.css("display", "none");
  restartPage.css("display", "flex");
});


// random
<html>
<head>
  <title> Meet the Beatles </title>
</head>
<body>
  <div id="band" class="content">
    <div id="leader1" class="leader ">
      John
      <div id="leader2" class="leader guitarist">Paul</div>
    </div>
    <div id="guitar1" class="guitarist">George</div>
    <div id="drummer">Ringo</div>
  </div>
  Dung
</body>
</html>
```