Week 3 (Sept 8–14)

Building Apps Accessible for Blind Users

Links for example:

- How Blind People Use Technology (My Apple Products - An Introduction to Voice …
- https://www.bairesdev.com/blog/how-to-build-a-handy-app-for-a-blind-person/

inks: How to use ARIA in HTML to make content accessible for blind users

- What is ARIA-Label and How to Use It Effectively | Web Accessibility Basics
- ARIA HTML Tutorial - What is ARIA & Why it's Important to Use!

Introduction
Making apps accessible for blind users means everyone can use your app.
Accessibility helps people who rely on screen readers and improves usability for Everyone.

1. Support Screen Readers
    - Add **alt text** to all images, icons, and buttons so screen readers can describe them.
    - Use **clear labels** for buttons and links (e.g., “Submit Form” instead of “Button 1”). For example, **Clear labels”** means giving buttons or links a descriptive name that tells the user exactly what will happen if they click it.
2. Make Navigation Easy
    - Users should be able to move around using **Tab, Enter, and arrow** keys.
    - Keep the **order of buttons and links logical** so it makes sense when reading with a screen reader.
    - On mobile, make **buttons large** and easy to tap, and give alternatives to gestures.
3. Keep Design Simple and Clear
    - Use **headings and simple layouts** so screen readers can understand the page.
    - Keep buttons and menus **consistent** across the app.
    - Use **short and clear instructions** for users.
4. Test your app / Websites
    - Try your app with **screen readers** like VoiceOver (iPhone), TalkBack (Android), or NVDA (Windows).

5. Extra Tips
    - Use **ARIA Labels** for custom buttons or interactive elements.
      Link: ARIA HTML Tutorial - What is ARIA & Why it's Important to Use!
      &
      What is ARIA-Label and How to Use It Effectively | Web Accessibilit…
    - Use **live updates** so screen readers announce changes on the page.
    - Keep forms **simple** with clear labels, instructions, and error messages.

Conclusion
Making your app accessible is important. It allows blind users to use it easily and improves the experience for everyone. Following these tips helps create apps that are inclusive, usable, and professional.

---

guidance on using semantic HTML

Links for more Information of semantic HTML for accessibility

- https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Accessibility/HTML
- https://www.a11y-collective.com/blog/html-screen-reader/

1. Use Sementic HTML
    - Use `<h1>`, `<h2>` for titles and sections.
    - Use `<button>` for buttons, not just `<div>`.
    - Use `<label>` with inputs so screen readers know what the box is for.
2. Keep the Design Simple
    - Break pages into clear sections with headings.
    - Put buttons and menus in the same spot on every page.
    - Write short, simple instructions (e.g., “Click Sumbit”).
3. Keyboard Navigation
    - Blind User dont always use a mouse. They use the **Tab** key to move and **Enter** to enter.
    - Make sure all buttons, links, and forms work with the keyboard.
4. Text-to-Speech (Screen Readers)
    - Sceen readers read out loud.
    - Add alt text to images so users know what the picture is.
      Example: `<img src="dog.jpg" alt="Golden retriever dog">`
5. Iphone voiceOver
    - Swipe right/left = move through items.
    - Double-tap = select the item.
    - Two fingers = scroll up/ down.
    - Three fingers = move a page at a time.

1. What is `aria-label`?

- `aria-label` is an HTML attribute that gives a **text description** to elements that might not have visible text.
- Screen readers use it to tell users what the element does.

2. When should we use it?

- Use it on **custom buttons or interactive elements** that don't have clear text.
- Example: a `<div>` used as a button, or an icon-only button.
- Helps blind users understand the purpose of elements.

3. Code Example:

❌ <!-- Custom button using a div (bad for accessibility without ARIA) -->

✅ <!-- Good: use aria-label so screen reader knows what it does -->

Week 4 (Sept 15–21)

To update accessibility checklist to include ARIA attributes for dynamic content and make notes about ensuring progressive enhancement if JavaScript fails.

1. Dynamic Content (ARIA)
Goal: Ensure screen readers announce updates automatically.

Announce text updates ➡ Sunny, 75°F
Urgent alers ➡ Serve weather warning!
Toggle menus/dropdowns ➡ <button aria-expanded= "False" aria-controls="menu1">Menu </button>

Modals / dialogs ➡ ...

2. Progressive Enhancement (if JavaScript fails)
Goal: Site must still work without JavaScript, if JavaScript fails.
Forms work without JavaScript ➡ <form action="/search" method="get">...</form>
Links work without JavaScript ➡ About
(Avoid href="#" as the only functionality.)
Core content visible ➡ <main>...</main> (Main content should load even if JavaScript fails.)

To test without Javascript, turn off Javascript Link:
https://www.youtube.com/watch?v=kBmvq2cE0D8

Week 5 — Text-to-Speech and Button Controls (Sept 22–28)

1. Concise Wording for Text-to-Speech

- Keep sentences short and direct.
  Example:
  - (Avoid this) “The weather forecast for today, which is Friday, October 5th, is expected to be sunny with some clouds.”
  - (Use this instead) “Today’s weather: sunny with some clouds.”

- Use plain words that are easy to pronounce
  Example:
  - “Temp.” ➡ “Temperature”
  - “75 F” ➡ “Seventy-five degrees Fahrenheit”

Put **key info first.**
Example:
- “Current temperature: 75 degrees. Mostly sunny.”

- Avoid unnecessary symbols (like ° or % in text-to-speech output).
- Use punctuation properly — it helps the voice pause naturally.

2. Best Voice Settings

- **Voice type:** Use a natural, calm voice — not too robotic or dramatic.
- **Pitch:** Keep it in the middle — not too high or too low.
- **Speed:** Talk at a normal speed so people can follow easily.
- **Emphasis:** Make important words stand out a little, like “temperature,” “rain,” or “wind.”
- **Gender:** You can use a male or female voice — just keep the same one for all messages.

Week 6 — Integrate API with Text-to-Speech, Add Error Handling, Deploy PoC (Sept 29–Oct 5)

1. Goal

- The goal is to test if the app is easy to use and if the accessibility features work well — like text-to-speech, buttons, and screen reader support.

2. Who Will Test

We will test with 6 people

- 2 blind or low-vision users who use screen readers
- 2 regular users
- 2 users with little tech experience

3. What We Will Use

- A phone or computer
- Screen readers like VoiceOver, TalkBack, or NVDA
- Internet connection for the weather API

4. What Testers Will Do

1. **Play the Weather Update** — Press the “Play” button and listen to the voice.
2. **Use Keyboard or Screen Reader** — Try to move around using Tab or swipe.
3. **Check Live Weather Info** — See if the app says new weather automatically.
4. **Test When Something Goes Wrong** — Turn off JavaScript or API and check if the app still works or shows a message.

5. What We Will Check

- **Can users understand what the voice says?**
- **Are the buttons easy to find and press?**
- **Do screen readers read everything correctly?**
- **How long does it take to finish each task?**
- **Are users happy with how it works?**

6. After the Test (Questions)

Ask users:

- **Was the voice clear?**
- **Was it easy to use the buttons?**
- **Did you understand the weather update?**
- **What would you change or improve?**

7. Success

We will know it works if:

- **Most users (about 90%) finish tasks without help.**
- **Everyone says the voice is clear.**
- **All buttons work with keyboard or screen reader.**
- **There are no big accessibility problems.**

Week 7 — Speech-to-Text Integration and Simplified Menus (Oct 6–12)

Test the speech-to-text feature on multiple browsers including Chrome, Firefox, and Safari.

Sample code to run the code for test to see if Chrome, Firefox, or Safari works & suportive

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Speech to Text Test</title>
</head>
<body>
 <h2>Speech to Text Demo</h2>
 <button id="start">Start Recording</button>
 <p id="result"></p>

 <script>
   const startBtn = document.getElementById("start");
   const result = document.getElementById("result");
   const SpeechRecognition = window.SpeechRecognition ||
window.webkitSpeechRecognition;

   if (!SpeechRecognition) {
     result.textContent = "Speech recognition not supported in this browser.";
   } else {
     const recognition = new SpeechRecognition();
     recognition.lang = "en-US";
     recognition.continuous = false;

     startBtn.addEventListener("click", () => recognition.start());
     recognition.onresult = (event) => {
       result.textContent = event.results[0][0].transcript;
     };
   }
 </script>
</body>
</html>
```

Test on these browsers

- Chrome - Works fully supportive ✅
- Safari - Works too as well ✅
- Firefox - Doesn't work, not supportive.

    - Document fallback strategies for browsers or devices that do not fully support the API.

Week 8 — Usability Testing and Polish (Oct 13–19)

Conduct user testing with blind or visually impaired users, summarize the results, and provide detailed recommendations for improvements.

Step 1 — Simulate a blind user,
Turn on a screen reader:

- On Mac: **VoiceOver**
- On Windows: **NVDA**
- On Chrome: use the **ChromeVox** extension

Step 2 – Test and see if it works

Everything worked well during the test:

- The screen readers correctly announced buttons, inputs, and headings.
- Weather information such as "Temperature: 75F and sunny" was spoken clearly
- Keyboard navigation worked smoothly — users could move between fields and buttons without a mouse.
- The speech-to-text feature also worked and gave accurate results.

Recommendations

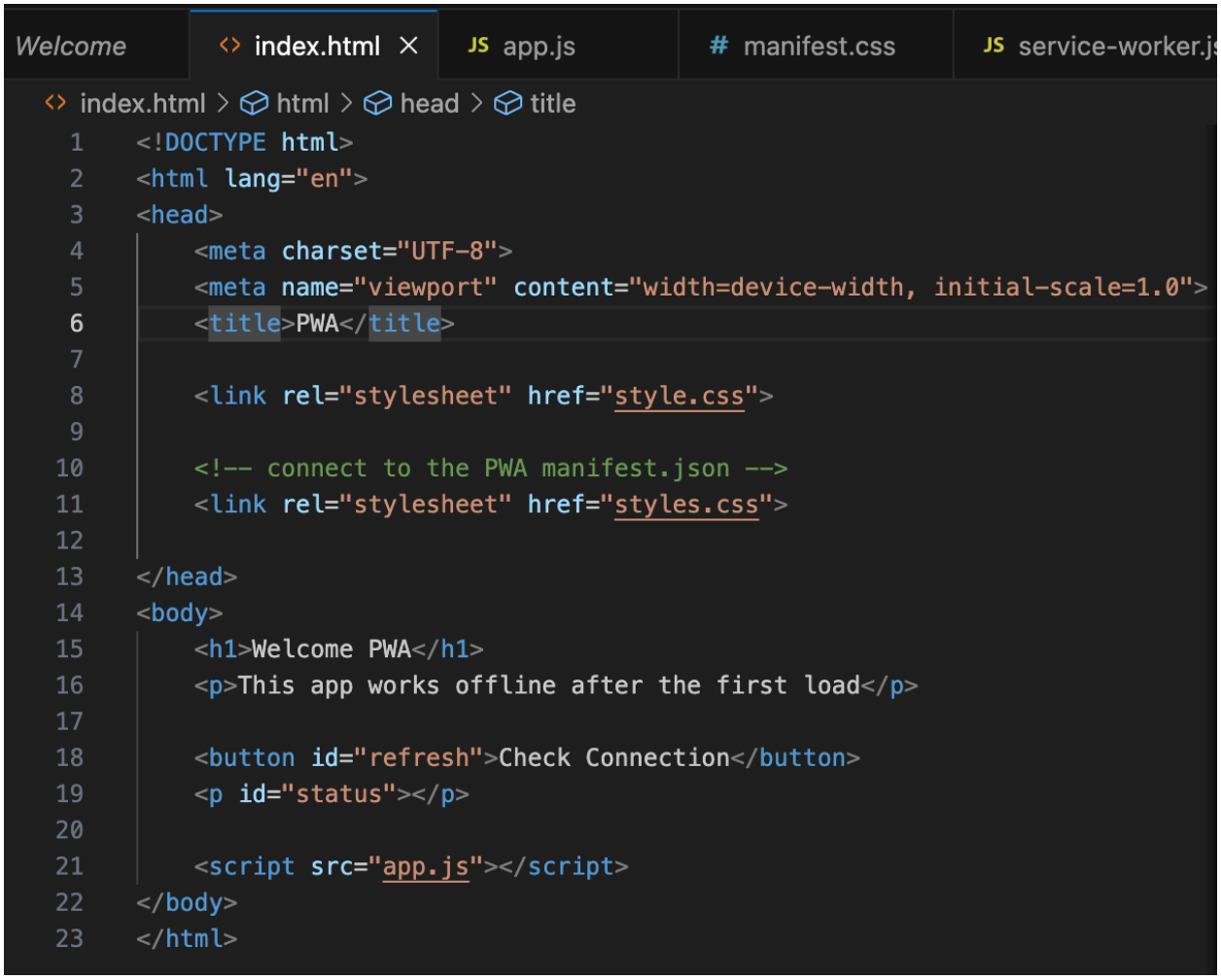
Even though everything worked, stull improve by:

1. Keeping all ARIA labels short and descriptive.
2. Testing on more browsers and devices to ensure consistent behavior.
3. Adding subtle focus outlines to help low-vision users.
4. Providing a short tutorial or voice hint for first-time users of the speech feature.

Week 10 — Offline Mode and Caching Optimization (Oct 27–Nov 2)

- Research accessibility best practices for Progressive Web Apps (PWAs)

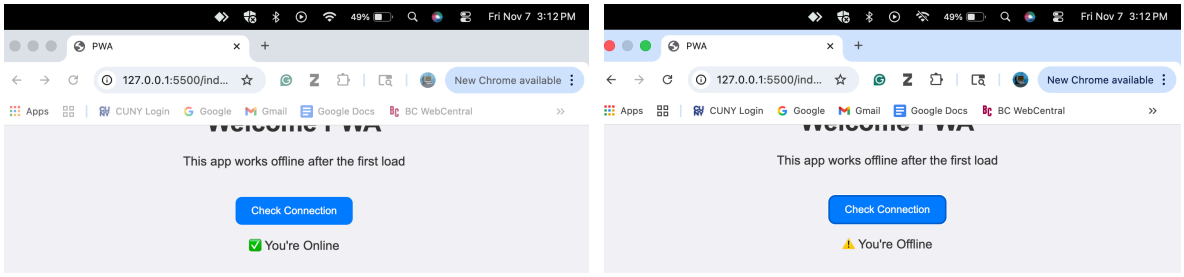
index.html



& app.js

```
//Regsiter the service worker
if('serviceWorker' in navigator){
   navigator.serviceWorker.register('service-worker.js')
   .then(()=> console.log('service worker Registered'))
   .catch(err => console.error('Service Worker Failed'));
}
// Simple online/offline check
document.getElementById('refresh').addEventListener('click', () => {
 const status = navigator.onLine ? "✅ You're Online" : "⚠️ You're Offline";
 document.getElementById('status').textContent = status; });
```

Online **vs** **Offline**



This picture shows the difference between when a **Progressive Web App (PWA)** is **online** and when it's **offline**.

When a PWA is **online**, it connects to the internet and loads live data, like new posts, messages, or updates. directly from the web server.

When the PWA is **offline**, it still **works without Wi-Fi** because it has already **saved (cached)** parts of the website on your device.

So even if the connection is lost, you can still open the app and see the content that was saved earlier.

The app can also show a **notification or alert** saying something like:

> "⚠️ You're offline. Some features may not be available."

Week 11 — User Settings and Personalization (Nov 3–9)

Week 12 — Prototype Integration and Internal Testing (Nov 10–16)

- Update the Accessibility Reference Document with new observations.

1. Navigation uses:
   - 🏠 Home

     & one button:

   - <button id="enable-voice-btn">🎙️ Enable Voice</button>

     Problems:

     Screen readers announce emoji strangely

     Mixed use of `<a>` and `<button>` inside navigation

     Buttons may not have consistent focus indicators

   - Recommendations:

     Use aria-label or write text beside emoji:

   - 🏠 Home

     Buttons should visually show outline when focused by Tab key.

2. Form Input & Labeling

- <input id="search" type="search" placeholder="City or ZIP"/>

  Problem:

  Input field has NO <label>

  Placeholder ≠ label (not accessible)

Recommendation:

Add: <label for="search">Enter location</label>

Why is a label important

Some users use screen readers because they are visually impaired or blind.

A screen reader **reads aloud** what's on the page.

Example when you have:

<input placeholder=”City or ZIP”>

THe screen reader might read: “Exit text blank” < which mean that doesnt tell the user WHAT to type.

But if there is a label:

- <label for="search">Enter location</label>
- <input id="search">

The Screen reader will read:

“Enter location –edit text”

Now the user knows what that field is for.

- Prepare formal user testing scripts and consent materials for the upcoming usability study.

1. Participant Consent Statement

Before beginning the session, read the following to the participant:

> Thank you for participating in this usability study. The purpose of this session is to understand how users interact with the product and identify areas for improvement.
> Your participation is voluntary, and you may stop at any time.
> We are evaluating the product, not you.
> Your responses will remain confidential and used only for research purposes.
> Do you consent to participate in this session?
> Do you also consent to audio/screen recording for analysis purposes?

Record the participant’s Yes/No.

1. What is your name and your current role/background?

   _____

2. Have you used similar apps or websites before? N\A _____
3. How often do you complete tasks like the ones we will test today?

   _____

4. What device do you normally use?

_____

5. Is there anything important I should know about how you use technology?

_____

3. Test Instructions

Read to participant:

> During this session, I will ask you to complete several tasks.
> There are no right or wrong answers — we are testing the design, not your skills.

4. Tasks for the Usability Test
The link of Websites: https://project4900.github.io/4900-Project-/index.html

Task 1: Visit the website using the link: https://project4900.github.io/4900-Project-/index.html

Task 2: Locate and explore the main navigation buttons

- **Weather**
- **Settings**
- **Exit**
- **Enable Voice**

Task 3: On the **Weather** page:

- **Enter a location by city or ZIP code**
- **OR select Use My Location**
- **View the weather details provided.**

Review the displayed weather information, including:

- **Current temperature**
- **Weather conditions**
- **5-day forecast**
- **Additional details (humidity, wind, etc.)**

Task 5:

Go to **Settings** and review all available options:

- **Dark Mode**
- **Big Text**
- **High Contrast**

- **Voice Settings** → Voice, Speed (recommended: 1), Pitch (recommended: 2)

Task 6: After completing the tasks, click **Exit**.

5. During-Task Probing Questions

- What are you expecting to happen here?

  _____

- What are you looking for right now?

  _____

- Is anything confusing?

  _____

- What would you do next if you were alone?

  _____

6. Post-Test Questions

1. How would you describe your overall experience?
2. What did you like the most?
3. Did you notice any errors, lagging, or miscommunication issues?
4. Was anything missing or unclear?
5. How confident did you feel during the tasks?
6. Would you use this product again? Why or why not?
7. Any suggestions for improvement?

	Week	Research	Links
1	Week 3	Accessible for blind users. —————————————————— guidance on using semantic HTML	How Blind People Use Tec… https://www.bairesdev.com/blog/how-to-build-a-handy-app-for-a-blind-person/ ARIA HTML Tutorial - Wha… —————————————————— https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Accessibility/HTML & https://www.a11y-collective.com/blog/html-screen-reader/
2	Week 4	**Progressive Enhancement (if JavaScript fails)**	Enable / Disable JavaScrip… & How to Disable or Enable …
3	Week 5	Text-to-Speech and Button Controls	Eleven Labs Best Voice S…
4	Week 6	Intergrate API with Text-to-Speech Add Error Handing, Deploy PoC (Week 6)	
5	Week 7	**Speech-to-Text Integration and Simplified Menus (Week 7)**	
6	Week 8		
7	Week 9		
8	Week 10	Research accessibility best practices for Progressive Web Apps (PWAs).	Introduction to PWAs [1 of 17]	P… & The Truth About Progressive We… & Progressive Web Apps in 100 Se…
9	Week 11		
10	Week 12		
11	Week 13		
1	Week 14		

1			
12	Week 15		
13	Week 16		