

Task 9: Implementing a Responsive UI

Objective:

Write JavaScript functions to adjust UI components based on screen size, implement performance optimizations, and enhance accessibility to ensure a responsive and user-friendly interface.

Pre-requisites:

- Basic understanding of HTML, CSS, and JavaScript
- Familiarity with a code editor like Visual Studio Code
- Basic knowledge of responsive design principles

Concepts Covered:

- Responsive Design with JavaScript
- Performance Optimization for Different Devices
- Accessibility Enhancements

Concepts:

1. Responsive Design with JavaScript:

Write JavaScript functions to adjust UI components based on screen size. Dynamically change styles or layout for mobile, tablet, and desktop views.

```
<div id="responsiveContainer">
  <h1>Responsive UI</h1>
  <p>This content adjusts based on screen size.</p>
</div>
```

```
function adjustLayout() {
  const container = document.getElementById('responsiveContainer');
  if (window.innerWidth < 600) {
    container.style.flexDirection = 'column';
    container.style.textAlign = 'center';
  } else if (window.innerWidth < 900) {
    container.style.flexDirection = 'row';
    container.style.textAlign = 'left';
  } else {
    container.style.flexDirection = 'row';
    container.style.textAlign = 'left';
  }
}

window.addEventListener('resize', adjustLayout);
document.addEventListener('DOMContentLoaded', adjustLayout);
```

2. Performance Optimization for Different Devices:

Implement lazy loading for images or content-heavy elements. Adjust JavaScript logic to enhance performance on mobile devices.

```


function lazyLoadImages() {
  const images = document.querySelectorAll('img[loading="lazy"]');
  const config = {
    rootMargin: '0px 0px 50px 0px',
    threshold: 0.01
  };

  let observer;

  if ('IntersectionObserver' in window) {
    observer = new IntersectionObserver((entries, self) => {
      entries.forEach(entry => {
        if (entry.isIntersecting) {
          preloadImage(entry.target);
          self.unobserve(entry.target);
        }
      });
    }, config);

    images.forEach(image => observer.observe(image));
  } else {
    images.forEach(image => preloadImage(image));
  }
}

function preloadImage(img) {
  const src = img.getAttribute('data-src');
  if (!src) return;
  img.src = src;
}

document.addEventListener('DOMContentLoaded', lazyLoadImages);
```

3. Accessibility Enhancements:

Ensure the website is accessible by adding keyboard navigation and screen reader support. Test and validate the accessibility of your website using tools like Lighthouse in Chrome DevTools.

```
<nav aria-label="Main Navigation">
  <ul>
    <li><a href="#home" tabindex="0">Home</a></li>
    <li><a href="#about" tabindex="0">About</a></li>
    <li><a href="#services" tabindex="0">Services</a></li>
    <li><a href="#contact" tabindex="0">Contact</a></li>
  </ul>
</nav>
```

```
function enableKeyboardNavigation() {
  const links = document.querySelectorAll('a');
  links.forEach(link => {
    link.addEventListener('keydown', event => {
      if (event.key === 'Enter') {
        link.click();
      }
    });
  });
}

document.addEventListener('DOMContentLoaded', enableKeyboardNavigation);
```

Setup:

1. Install Visual Studio Code (VS Code):

Download and install VS Code from [Visual Studio Code](#).

2. Web Browsers:

Use Google Chrome or Mozilla Firefox for viewing your webpage and utilizing their developer tools for debugging.

Tasks:

1. Responsive Design with JavaScript (10 minutes):

- Write JavaScript functions to adjust UI components based on screen size.
- Dynamically change styles or layout for mobile, tablet, and desktop views.
- Example:

```
<div id="responsiveContainer">
  <h1>Responsive UI</h1>
  <p>This content adjusts based on screen size.</p>
</div>
```

```
function adjustLayout() {
  const container = document.getElementById('responsiveContainer');
  if (window.innerWidth < 600) {
    container.style.flexDirection = 'column';
    container.style.textAlign = 'center';
  } else if (window.innerWidth < 900) {
    container.style.flexDirection = 'row';
    container.style.textAlign = 'left';
  } else {
    container.style.flexDirection = 'row';
    container.style.textAlign = 'left';
  }
}

window.addEventListener('resize', adjustLayout);
document.addEventListener('DOMContentLoaded', adjustLayout);
```

2. Performance Optimization for Different Devices (10 minutes):

- Implement lazy loading for images or content-heavy elements.
- Adjust JavaScript logic to enhance performance on mobile devices.
- Example:

```

```

```
function lazyLoadImages() {
  const images = document.querySelectorAll('img[loading="lazy"]');
  const config = {
    rootMargin: '0px 0px 50px 0px',
    threshold: 0.01
  };

  let observer;

  if ('IntersectionObserver' in window) {
    observer = new IntersectionObserver((entries, self) => {
      entries.forEach(entry => {
        if (entry.isIntersecting) {
          preloadImage(entry.target);
          self.unobserve(entry.target);
        }
      });
    }, config);

    images.forEach(image => observer.observe(image));
  } else {
    images.forEach(image => preloadImage(image));
  }
}
```

```
function preloadImage(img) {
  const src = img.getAttribute('data-src');
  if (!src) return;
  img.src = src;
}

document.addEventListener('DOMContentLoaded', lazyLoadImages);
```

3. Accessibility Enhancements (10 minutes):

- Ensure the website is accessible by adding keyboard navigation and screen reader support.
- Test and validate the accessibility of your website using tools like Lighthouse in Chrome DevTools.
- Example:

```
<nav aria-label="Main Navigation">
  <ul>
    <li><a href="#home" tabindex="0">Home</a></li>
    <li><a href="#about" tabindex="0">About</a></li>
    <li><a href="#services" tabindex="0">Services</a></li>
    <li><a href="#contact" tabindex="0">Contact</a></li>
  </ul>
</nav>
```









```
function enableKeyboardNavigation() {
  const links = document.querySelectorAll('a');
  links.forEach(link => {
    link.addEventListener('keydown', event => {
      if (event.key === 'Enter') {
        link.click();
      }
    });
  });
}

document.addEventListener('DOMContentLoaded', enableKeyboardNavigation);
```

Instructions:

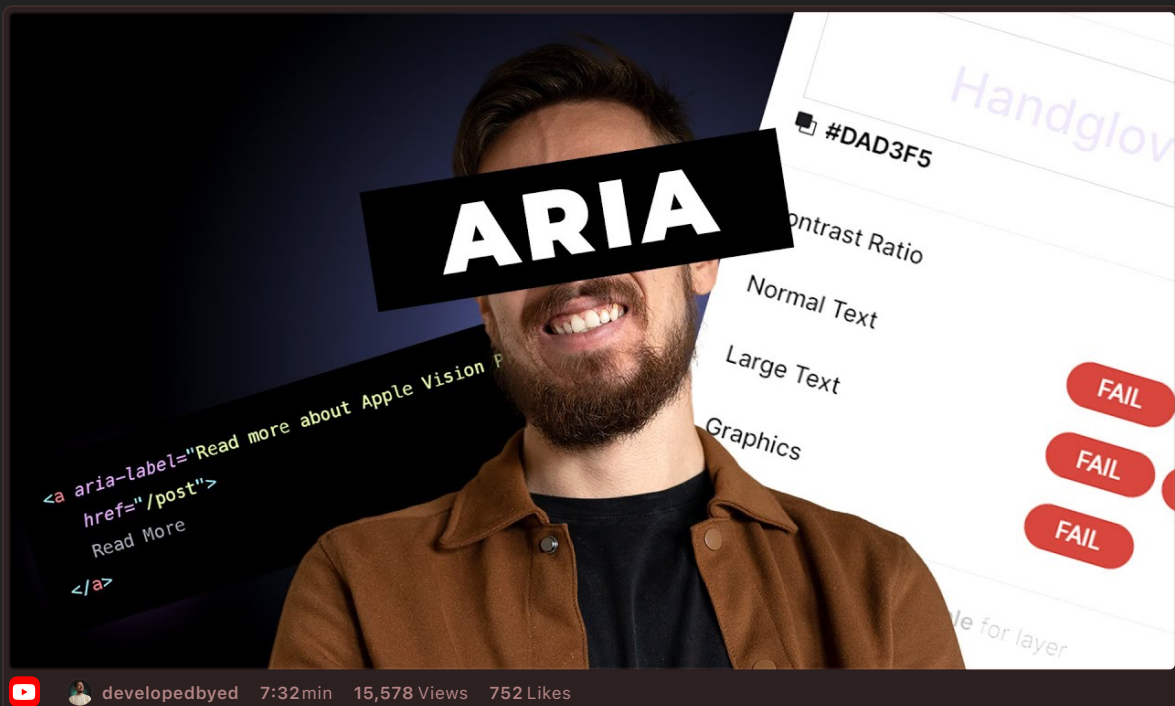
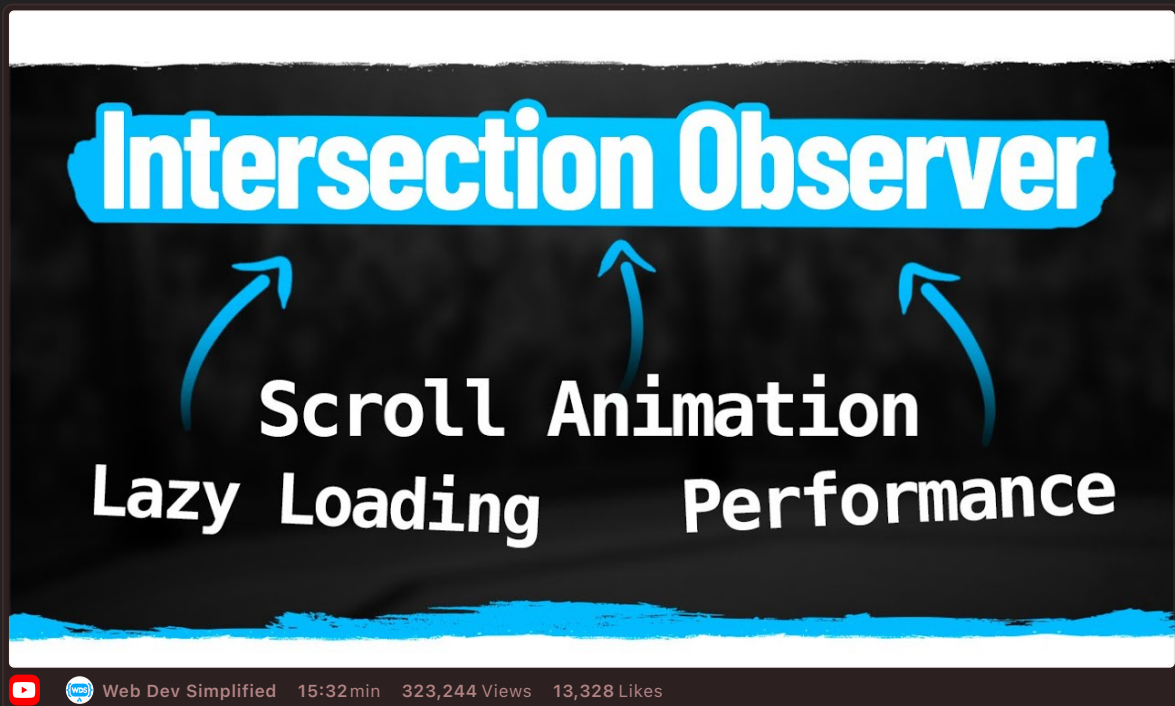
1. Write the required code in `index.html` and `script.js`.
2. Open the `index.html` file in your web browser to ensure the code displays correctly.
3. Use the browser's developer tools to debug and inspect the elements.

Resources:

-  **Responsive design - Learn web development | MDN**
Responsive design refers to a site or application design that responds to the environment i...
 https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design
-  **Intersection Observer API - Web APIs | MDN**
The Intersection Observer API provides a way to asynchronously observe changes in the in...
 https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API
-  **ARIA - Accessibility | MDN**
Accessible Rich Internet Applications (ARIA) is a set of roles and attributes that define way...
 <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA>
-  **Documentation for Visual Studio Code**
Find out how to set-up and get the most from Visual Studio Code. Optimized for building and d...
 <https://code.visualstudio.com/docs>

Videos:

- 



GitHub Instructions:

1. Open in Visual Studio Code:

After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, VSCode will open the repository directly. If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. Open the Terminal in VSCode:

In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

3. Complete the Task:

In VSCode, write your solution in the `index.html` and `script.js` files.

4. Run and Test Your Code:

Open your `index.html` file in a web browser to ensure it works correctly. Use the following command:

```
open index.html
```

5. Commit Your Changes:

In the VSCode terminal, add your changes to git:

```
git add index.html script.js
```

Commit your changes with a meaningful message:

```
git commit -m "Completed task 20"
```

6. Push Your Changes to Your Repository:

Push your changes to your forked repository:

```
git push origin main
```

7. Create a Pull Request:

Go to your repository on GitHub.

Click on the "Pull Requests" tab.

Click the "New Pull Request" button.

Ensure the base repository is the original template repository and the base branch is `main`.

Ensure the head repository is your forked repository and the compare branch is `main`.

Click "Create Pull Request".

Add a title and description for your pull request and submit it.

Summary of Commands:


```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing index.html and script.js

# Navigate to the directory containing index.html
cd path/to/your/index.html

# Run your code
open index.html

# Add, commit, and push your changes
git add index.html script.js
git commit -m "Completed task 9"
git push origin main

# Create a pull request on GitHub
```