**HEADER**

**First part of html**

**Step 1**

Welcome to the first part of the Accessibility Quiz. As you are becoming a seasoned HTML and CSS developer, we have started you off with the basic boilerplate.

Start this accessibility journey by providing a lang attribute to your html element. This will assist screen readers in identifying the language of the page.

<html lang="en">

  <head>

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

  </head>

  <body>

  </body>

</html>

# Step 2

You may be familiar with the meta element already; it is used to specify information about the page, such as the title, description, keywords, and author.

Give your page a meta element with an appropriate charset value.

The charset attribute specifies the character encoding of the page, and, nowadays, UTF-8 is the only encoding supported by most browsers.

  <head>

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

    <meta charset="utf-8">

  </head>

# Step 3

Continuing with the meta elements, a viewport definition tells the browser how to render the page. Including one betters visual accessibility on mobile, and improves SEO (search engine optimization).

Add a viewport definition with a content attribute detailing the width and initial-scale of the page.

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

# Step 4

Another important meta element for accessibility and SEO is the description definition. The value of the content attribute is used by search engines to provide a description of your page.

Add a meta element with the name attribute set to description, and give it a useful content attribute.

<meta name="description" content="Free Web tutorials">

# Step 5

Lastly in the head, the title element is useful for screen readers to understand the content of a page. Furthermore, it is an important part of SEO.

Give your page a title that is descriptive and concise.

<title> Sohaib Hassan Quizz</title>

# Step 6

Navigation is a core part of accessibility, and screen readers rely on you to provide the structure of your page. This is accomplished with semantic HTML elements.

Add a header and a main element to your page.

The header element will be used to introduce the page, as well as provide a navigation menu.

The main element will contain the core content of your page.

<header>

      </header>

    <main>

      </main>

# Step 7

Within the header, provide context about the page by nesting one img, h1, and nav element.

The img should point to https://cdn.freecodecamp.org/platform/universal/fcc\_primary.svg, and have an id of logo.

The h1 should contain the text HTML/CSS Quiz.

<header>

      <!-- context add here -->

      <img src="https://cdn.freecodecamp.org/platform/universal/fcc\_primary.svg" id="logo"/>

      <h1>HTML/CSS QUIZZ</h1>

      <nav></nav>

    </header>

**Step 8**

A useful property of an *SVG* (scalable vector graphics) is that it contains a path attribute which allows the image to be scaled without affecting the resolution of the resultant image.

Currently, the img is assuming its default size, which is too large. CSS has a max function which returns the largest of a set of comma-separated values. For example:

img {

width: max(250px, 25vw);

}

In this example, img elements will have a minimum width of 250px. And as the viewport grows, the image will grow accordingly to be 25 percent of the viewport width.

Scale the image using its id as a selector, and setting the width to be the maximum of 100px or 18vw.

#logo{

  width:max(100px,18vw);

}

Here we will use a function max to resize our image

# Step 9

As described in the [freeCodeCamp Style Guide](https://design-style-guide.freecodecamp.org/), the logo should retain an aspect ratio of 35 / 4, and have padding around the text.

First, change the background-color to #0a0a23 so you can see the logo. Then, use the aspect-ratio property to set the desired aspect ratio to 35 / 4. Finally, add a padding of 0.4rem all around.

#logo {

  width: max(100px, 18vw);

  background-color:#0a0a23;

  aspect-ratio:35/4;

  padding:0.4rem;

}

Note

Here we will design a logo there are a website fot free code camp org which well have a lot of resourses in it about logo to layout even colors everything is available just need your patient and be calm try to visit freecodecamp org style website Thank you!

The link is provided here

[freeCodeCamp Style Guide](https://design-style-guide.freecodecamp.org/),

# Step 10

Make the header take up the full width of its parent container, set its height to 50px, and set the background-color to #1b1b32. Then, set the display to use Flexbox.

Here we will use a Flexbox property which is very important for understanding in our case it should be a starting let strt

Be calm

header{

  height:50px;

  background-color:#1b1b32;

  display:flexbox;

}

header{

  width:100%;

  height:50px;

  background-color:#1b1b32;

  display:flex;

}

Here we are dealing with header first we design a header then move further

For header it should be a maxumun height of 50px

Bacgroundcolor black

Display for container is flexbox =display:flex;

Here we make a mistake forget width width should must be define with height it will take a full width of the screen

# Step 11

Change the h1 font color to #f1be32, and set the font size to min(5vw, 1.2em).

Now we are dealing with h1 mean inside header there is an h1 element we design it using different techniques

Min(5vw, 1.2em)

Remember

1em =10px so 12em = 12\*10 = 120px;

h1{

  color:#f1be32;

  font-size:min(5vw, 1.2em);

}

"Happy, happy, joy, joy!"

"Happy, happy, joy, joy!"

# Step 12

To enable navigation on the page, add an unordered list with the following three list items:

* INFO
* HTML
* CSS

The list items text should be wrapped in anchor tags.

<nav></nav>

For nav element we should include unorder list

     <nav>

<ul>

  <li><a href="#">INFO</a></li>

    <li><a href="#">HTML</a></li>

      <li><a href="#">CSS</a></li>

      </nav>

NOTE

WE USE A NAV INSDE THE HEADER

"Nice one!"

# Step 13

Use the > selector to target the unordered list elements within the nav elements, and use Flexbox to evenly space the children.

Here we will learn a new topic which is use > this symbol and are called children selector

nav > ul{

  display:flex;

}

Now this will select all the child of nav and inside ul tags which is label in our case

nav > ul{

  display:flex;

  justify-content:space-evenly;

  color:white;

}

Note flex is a container

It is row by default e also set a column felx direction

Justify content space-even is a space between element it have different property

# Step 14

As this is a quiz, you will need a form for users to submit answers. You can semantically separate the content within the form using section elements.

Within the main element, create a form with three nested section elements. Then, make the form submit to https://freecodecamp.org/practice-project/accessibility-quiz, using the correct method.

Here I will practice on it from start

Lets leave it for now go to another page for main element

It will first task note it will do 5 times minimum

Main Element

After header this part come

We have use section in main element mainly

Let’s start

<main>

<section>

</section>

</main>

This is the second part of html structure we add many section according to our need and demand

Let’s start

# Step 14

As this is a quiz, you will need a form for users to submit answers. You can semantically separate the content within the form using section elements.

Within the main element, create a form with three nested section elements. Then, make the form submit to https://freecodecamp.org/practice-project/accessibility-quiz, using the correct method.

  <main>

      <form action="https://freecodecamp.org/practice-project/accessibility-quiz" method="post"/>

<section></section>

<section></section>

<section></section>

</form>

    </main>

Okay first thing first we make a three section inside a main element with in main element we create a form

There are two things important

Method: use for post

Action: use for where to submit our form or data

# Step 15

To increase the page accessibility, the role attribute can be used to indicate the purpose behind an element on the page to assistive technologies. The role attribute is a part of the Web Accessibility Initiative (WAI), and accepts preset values.

Give each of the section elements the region role.

   <section role="region"></section>

        <section role="region"></section>

        <section role="region"></section>

Now you can see here we add a Role=”region”

So basically it is important for web accessibility initiative(WAI)

Accessibility mean

The [quality](https://dictionary.cambridge.org/dictionary/english/quality) of being [able](https://dictionary.cambridge.org/dictionary/english/able) to be [entered](https://dictionary.cambridge.org/dictionary/english/enter) or used by everyone, [including](https://dictionary.cambridge.org/dictionary/english/include) [people](https://dictionary.cambridge.org/dictionary/english/people) who have a [disability](https://dictionary.cambridge.org/dictionary/english/disability):

**Step 16**

Every region role requires a label, which helps screen reader users understand the purpose of the region. One method for adding a label is to add a heading element inside the region and then reference it with the aria-labelledby attribute.

Add the following aria-labelledby attributes to the section elements:

* student-info
* html-questions
* css-questions

Then, within each section element, nest one h2 element with an id matching the corresponding aria-labelledby attribute. Give each h2 suitable text content.

 <form method="post" action="https://freecodecamp.org/practice-project/accessibility-quiz">

        <section role="region" aria-labelledby="student-info">

          <h2 id="student-info"> Student Info</h2>

        </section>

       <section role="region" aria-labelledby="html-questions">

          <h2 id="html-questions">Html Questions</h2>

        </section>

  <section role="region" aria-labelledby="css-questions">

          <h2 id="css-questions">CSS Questions</h2>

        </section>

      </form>

"Boom-shakalaka!"