Thursday Jan 25th 2024

[Accessibility - Building a Quiz]

① (index.html) <html lang="en">

② (index.html) → <meta charset="utf-8">

③ (index.html) → <meta name="viewport" content="width=device-width, initial-scale=1">

④ (index.html) Another important meta element for accessibility ~~H/A~~ and SEO is the description definition. The value of the content attribute is used by search engines to provide a description of your page.

<meta name="description" content="useful">

⑤ (index.html) important part of SEO → <title></title>

add the title → <title>Quiz practice project</title>

⑥ (index.html)

<header></header> → to introduce the page, as well as provide a navigation menu.

<main></main> → will contain the core content of your page.

⑦ (index.html)

```
<header>
  <img id="logo" src="https://....jpg">
  <h1>HTML/CSS Quiz</h1>
  <nav></nav>
</header>
```

→ it should be .svg

⑧ (styles.css) 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. ex:

```
img {
  width: max(250px, 25vw);
}
```

→ and as the viewport grows, the image will grow acc

↓ minimum 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);
}
```

⑨ (styles.css) the logo should retain an aspect ratio of 35 / 4.

[cont'd]

```
background-color : #0a0a23;
aspect-ratio : 35 / 4 ;
padding : 0.4rem;
```

⑩ (styles.css) for .header →

```
width: 100%;
height: 50px;
background-color : #161632;
display : flex;
}
```

⑪ (styles.css)

```
h1 {
  color : #f1be32;
  font-size : min (5vw, 1.2em);
}
```

⑫ (index.html) Enabling navigation on the page, adding unordered list following three list items : INFO, HTML, CSS. And wrap these 3 in anchor tags.

```
<nav>
  <ul>
    <li><a>INFO</a></li>
    <li><a>HTML</a></li>
    <li><a>CSS</a></li>
  </ul>
</nav>
```

⑬ (styles.css)

The child combinator selector > is used between selectors to target only elements that match the 2nd selector and are a direct child of 1st selector.

This can be helpful when you have deeply nested elements and want to control the scope of your styling.

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

```
nav > ul {
  display: flex;
  justify-content : space-evenly;
}
```

(14) (index.html) [READ CAREFULLY!]

1) Separate the content within the form using section elements.
2) Within the main element, create a form with three nested section elements.
3) Use the correct method to submit the form.

```
<main>
  <form action="(link)" method="POST">
  <section></section> → <section role="region"></section>
  <section></section> → <section role="region"></section>
  <section></section> → <section role="region></section>
  </form>
</main>
```

STEP 14 + 15

(15) (index.html) [READ CAREFULLY!]

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 accept preset values.

give each section elements the region role (see step 14).

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

(16) (index.html) [READ CAREFULLY!]

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-labelledly attribute.

add the following aria-labelledly 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-labelledly attribute. Give each h2 suitable text content.

(17) (styles.css) READ CAREFULLY!

Typeface plays an important role in the accessibility of a page. Some fonts are easier to read than others, and this is especially true on low resolution screens.

changing the fonts to Verdana. and use sans-serif family as a fallback. BUT add border-bottom for h2.

```
h2, h1 {
  font-family: Verdana, sans-serif;
}
```

```
h2 {
  border-bottom: 4px solid #dfdfe2;
}
```

(18) (index.html) Giving an anchor element an href corresponding to the id of h2 elements.

```
<li><a href="#student-info">INFO</a></li>
<li><a href="#html-questions">HTML <a></li>
```

(19) (index.html)

[INI DIISINYA DI DALEM SECTION student-info]

```
<div class="info"><label></label><input></div>  x3.
```

(20) (index.html) READ CAREFULLY!

It is important to link each input to the corresponding label element. This provides assistive technology users with a visual reference to the input. This is done by giving the label a for attribute, which contains the id of the input. This section will take a student's name, email address, and D.O.B. Give the label elements appropriate for attributes, as well as text content. Then, link the input elements to the corresponding label elements.

SERING KETINGGALAN

Format example :

```
<div class="info">
  <label for="student's-name">Name</label>
  <input id="student's-name">
</div>
```

* value attribute for & id harus sama!!!

(21) (index.html) Keeping in mind best-practices for from inputs, give each input an appropriate type & name attribute. Then, give the first input a placeholder attribute.

READ CAREFULLY !

```
<section role="region" aria-labelledby="student-info">
  <h2 id="student-info">Student Info</h2>
  <div class="info">
    <label for="student-name">Name:</label>
    <input id="student-name" type="text" name="name" placeholder="Enter
      your name">
  </div>
  <div class="info">
    <label for="student-email">Email:</label>
    <input id="student-email" type="email" name="email" placeholder="En-
    ter your email">
  </div>
  <div class="info">
    <label for="birth-date">D.O.B:</label>
    <input id="birth-date" type="date" name="date" placeholder="Enter
      your date of birth (DDMMYYYY)">
  </div>
```

WAJIB !

JGN LUPA !!!

WAJIBUN !

(22) (index.html) even though you added a placeholder to the first input element in the previous lesson, this is actually not a best-practice for accessibility; too often, users confuse the placeholder text with an actual input value - they think there is already a value in the input.

remove the placeholder text from the first input element, relying the label being the best-practice.

cuma dihapus bagian placeholdernya aja.

(23) (index.html) arguably, D.O.B is not descriptive enough. This is especially true for visually impaired users. One way to get around such an issue, without having to add visible text to the label, is to add text only a screen reader can read.

Append a span element with a class of sr-only to the current text content of the label element.

```
<div class = "info">
  <label for = "birth-date"> D.O.B. <span class = "sr-only"></span></label>
  <input type = "date" name = "birth-date" id = "birth-date">
</div>
```

(24) (index.html) adding text (Date of Birth) within the span element [see step 23].

(25) (styles.css) defining the sr-only class with provided CSS properties.

```
.sr-only { }
```

(26) (index.html) adding <fieldset class = "question"></fieldset> within the element.

(27) (index.html) Each fieldset will contain a true/false question. Within each fieldset, nest one legend element, and one ul element with 2 options.

```
ex: <fieldset class = "question"><legend></legend><ul><li></li><li></li></ul>
```

(28) (index.html) giving each fieldset an adequate name attribute. Then, give both unordered lists a class of answers-list. Use the legend to caption the content of the fieldset by placing a true/false question as the text content.

```
<field class = "question" name = "html-1">
  <legend> xxx? </legend>
  <ul class = "answers-list">
    <li></li>
    <li></li>
  </ul>
```

terserah kita mau ditulis apa

ini juga up to you

(29) (index.html) to provide the functionality of the true/false questions, we need a set of inputs which do not allow both to be selected at the same time.

within each list element, nest one label element, and within each label element, nest one input element with the appropriate type.

```
<li><label><input type = "radio"></label></li>
```

(30) (index.html) Adding an id to all of your radio input so you can link your labels to them.

Give the 1st one an id of q1-a1. Give the rest of them ids of q1-a2, q2-a1, and q2-a2, respectively.

ex: <input type="radio" id="q1-a1">
<input type="radio" id="q1-a2">
<input type="radio" id="q2-a1">
<input type="radio" id="q2-a2">

(31) (index.html) Although not required for label elements with a nested input, it is still best-practice to explicitly link a label with its corresponding input element. Now, add a for attribute to each of your four labels that links the label to its corresponding radio input.

ex: <label for="q1-a2">
<input type="radio" id="q1-a2">
</label>

maybe the value of "for" and "id" should be same.

(32) (index.html) give the label elements text such that the input comes before the text. Then, give the input elements a value matching the text. The text should either be true or false.

ex: <label for="q1-a2"><input type="radio" id="q1-a2" value="False">False
</label>

bisa diganti false/true.

(33) (index.html) [Biar radio buttonnya cuma bisa diklik salah satunya aja,

Pake format ini:

<label for="q1-q1"><input type="radio" id="q1-a1" value="true" name="true">
True </label>

<label for="q1-a2"><input type="radio" id="q1-a2" value="false" name="true">
False</label>

harus nama value namenya

REMEMBER !!!

(34) (styles.css) to prevent unnecessary repetition, target the before pseudo-element of the p element, and give it a content property of "Question #".

```
p::before {
  content: "Question #";
}
```

(35) (index.html) adding dropdown, and a text box.

```
<div class="formrow">
<div class="question-block"></div>
<div class="answer"></div>
<div class="question-block"></div>
<div class="answer"></div>
</div>
```

(36) (index.html) within the div.question-block elements, nest one label element, and add a CSS related question to the label text.

```
<div class="question-block">
  <label>What is CSS?</label>
</div>
```

} just an example

(37) (index.html) within the first div.answer element, nest one required select element with three option elements.

#1 give the 1st option element a value of " ", and the text Select, as an option.

#2 give the 2nd option element a value of yes + the text Yes

#3 give the 3rd option element a value of no + the text No

→

```
<div class="answer">
  <select required>
  <option value=" ">Select an option</option>
  <option value="yes">Yes</option>
  <option value="no">No</option>
  </select>
</div>
```

→ PENTING BGT !!!

(38) (index.html) Linking the 1st label element to the select element, and give the select element a name attribute

PLS REMEMBER !! → kalau disuruh nge-link-in tiap elemen, harus ada for attribute, id attribute, dan name attribute.

Ex: <label for="genius"> Are you happy?</label>
<select required id="genius" name="ingus"></select>
→ aku namainnya ngasal"

note: value for dan value id harus disamain !

(39) (index.html) nest one textarea element within the 2nd div.answer element, and set the number of rows and columns it has. Then, give the textarea placeholder text describing an example answer.

```
<div class="answer">
  <textarea rows="2" cols="2" placeholder="ingus"></textarea>
</div>
```

rows: ingetin bentuknya begini

cols: "2"

placeholder="ingus": ini valuenya aku bikin sendiri ☺

(40) (index.html)

```
<div class="question-block">
  <label for="abc"> Any questions?</label>
</div>
<div class="answer">
  <textarea rows="5" cols="24" id="abc" name="ajuu" placeholder="x">
  </textarea>
</div>
```

ngasal

ngasal

(41) (index.html) giving your form a submit button with the text Send

[ternyata, button dan ~~input~~ label itu hampir sama]

```
<button type="submit">Send</button>
```

(42) (index.html) adding footer & address

```
<footer>
  <address></address>
</footer>
```

added after the main element.

(43) (index.html)
Within the address, add the following:

```
freeCodeCamp <br />
San Fransisco <br />
California <br />
USA
```

} the br tags will allow each part of the address to be on its own line and are useful for presenting address elements properly

(44) (index.html)

```
<a href="txvyz.org">freeCodeCamp</a><br />
```

(45) (styles.css) styling the page. Select the list elements within the navigation bar and give following styles:

```
nav li {
  color: #dfdfe2;
  margin: 0 0.2rem;
  padding: 0.2rem;
  display: block;
}
```

(46) (styles.css) on the topic of visual accessibility, contrast between elements is a key factor. For example, the contrast between the text and the background of a heading should be at least 4.5:1

```
li > a {
  color: #dfdfe3
}
```

(47) (styles.css)

```
nav li:hover {
  background-color: #dfdfe2;
  color: #161632;
  cursor: pointer;
}

li > a {
  color: inherit;
  text-decoration: none;
}
```

(48) (styles.css)

READ CAREFULLY!

1) Tidy up the header by using flexbox to put space between the children, and vertically center them.
2) Fix the header to the top of the viewport

```
header {
  width: 100%;
  height: 50%;
  background-color: #161632;
  display: flex;
  justify-content: space-between
  align-items: center;
  position: fixed
  top: 0;
```

(49) (styles.css)

```
main {
  padding-top: 25px;
}
```

(50) (styles.css) Fixing the appearance using Flexbox.

```
nav > ul {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-evenly;
  align-items: center;
  padding-inline-start: 0;
  margin-block: 0;
  height: 100%;
}
```

(51) (styles.css) Read the instruction carefully!

1) set the width of the section elements → 80%.
2) use margins to center the sections → margin-left: auto;
   margin-right: auto;
3) set the bottom margin 10px → margin-bottom: 10px;
4) ensure the section elements CANNOT BE LARGER than 600px in width
   ↳ max-width: 600px;

(52) (styles.css) replacing the top margin & adding a padding.

```
h2 {
  border-bottom: 4x solid #dfdfe2;
  Padding-top: 60px;
  margin-top: 0 px;
}
```

(53) (styles.css)

```
.info {
  Padding-top: 2px;
  Padding-bottom: 0px;
  Padding-right: 0 px;
  Padding-left: 2px;
}
```

(54) (styles.css)

```
.formwor {
  margin-top: 2px;
  Padding-top: 0 px;
  Padding-bottom: 0 px;
  Padding-right: 2px;
  Padding-left: 2 px;
}

input {
  font-size: 16px;
}
```

55 (styles.css) READ THE INSTRUCTIONS CAREFULLY!

To make the 1st section look more inline, target only the input elements within .info elements, and set their width to 50% and left-align their text.

```
.info > input {
  width : 50%;
  text-align : left;
}
```

56 (styles.css) Target all label elements within .info elements, and set their width to 10%, and make it so they do not take up less than 55px.

```
.info > label {
  width : 10%;
  min-width : 55px;
}
```

ini bisa juga ditulis :

```
.info label {}
```

57 (styles.css)

↓

a
```
.info input {
  width : 50%;
  text-align : left;
}
```

b
```
.info label {
  width : 10%;
  min-width : 55px;
}
```

```
.info input, .info label {
  display : inline-block;
  text-align : right;
}
```

58 (styles.css) modifying the .question-block elements

```
.question-block {}
```

60 (styles.css)
```
.question {
  border : none;
  padding-bottom : 0;
}
```

59 (styles.css)
```
p {
  margin-top : 5px;
  padding-left : 15px;
  font-size : 20px;
}
```

61
```
.answers-list {
  list-style : none;
  padding : 0;
}
```

↓

styles.css

62 (styles.css) setting the submit button.

63 (styles.css) setting the footer.

```
footer {
  background-color : #2a2a40;
  display : flex;
  justify-content : center;
}
```

(64) (styles.css)

```
footer, footer a {
    color : #dfdfe2;
}
```

(65) (styles.css)

```
address {
    padding-top : 2px;
    :
    :
    text-align : center;
}
```

(66) (styles.css)

```
* {
scroll-behavior : smooth;
}
```

→ universal selector

(67) (styles.css) certain types of motion-based animations can cause discomfort for some users. In particular, people with vestibular disorders have sensitivity to certain motion triggers.

The media (using @) can (take) one of the following values:

- reduce
- no-preference.

Syntax →

```
@media (feature : value) {
    selector {
        styles
    }
}
```

```
@media (prefers-reduced-motion:
no-preference) {
* {
scroll-behavior : smooth;
}
}
```

(68) (index.html) Adding accesskey attribute (providing keyboard shortcuts).

This attribute a space-separated list of access key. Ex:

<button type="submit" accesskey="s">Submit</button>

give each of the navigation links a single-letter access key.

(Note: it is not always advised to use access keys, but they can be useful).