Content

- **Alt use in <img>**
- **Audio in html**
- **Fieldset**
- **Date-picker**
- **Datetime attribute**
- **AccessKey**
- **label**
- **tabindex**

Applied Accessibility: Know When Alt Text Should be Left BlankPassed

In the last challenge, you learned that including an alt attribute when using img tags is mandatory. However, sometimes images are grouped with a caption already describing them, or are used for decoration only. In these cases alt text may seem redundant or unnecessary.

In situations when an image is already explained with text content, or does not add meaning to a page, the img still needs an alt attribute, but it can be set to an empty string. Here's an example:

```
<img src="visualDecoration.jpeg" alt="">
```

Background images usually fall under the 'decorative' label as well. However, they are typically applied with CSS rules, and therefore not part of the markup screen readers process.

Note: For images with a caption, you may still want to include alt text, since it helps search engines catalog the content of the image.

==========

Applied Accessibility: Improve Accessibility of Audio Content with the audio ElementPassed

HTML5's audio element gives semantic meaning when it wraps sound or audio stream content in your markup. Audio content also needs a text alternative to be accessible to people who are deaf or hard of hearing. This can be done with nearby text on the page or a link to a transcript.

The audio tag supports the controls attribute. This shows the browser default play, pause, and other controls, and supports keyboard functionality. This is a boolean attribute, meaning it doesn't need a value, its presence on the tag turns the setting on.

Here's an example:

```
<audio id="meowClip" controls>
  <source src="audio/meow.mp3" type="audio/mpeg" />
  <source src="audio/meow.ogg" type="audio/ogg" />
</audio>
```

Note: Multimedia content usually has both visual and auditory components. It needs synchronized captions and a transcript so users with visual and/or auditory impairments can access it. Generally, a web developer is not responsible for creating the captions or transcript, but needs to know to include them.

==========

Applied Accessibility: Wrap Radio Buttons in a fieldset Element for Better Accessibility

The next form topic covers accessibility of radio buttons. Each choice is given a label with a for attribute tying to the id of the corresponding item as covered in the last challenge. Since radio buttons often come in a group where the user must choose one, there's a way to semantically show the choices are part of a set.

The fieldset tag surrounds the entire grouping of radio buttons to achieve this. It often uses a legend tag to provide a description for the grouping, which is read by screen readers for each choice in the fieldset element.

The fieldset wrapper and legend tag are not necessary when the choices are self-explanatory, like a gender selection. Using a label with the for attribute for each radio button is sufficient.

Here's an example:

```
<form>
  <fieldset>
    <legend>Choose one of these three items:</legend>
```

```
  <input id="one" type="radio" name="items" value="one">
  <label for="one">Choice One</label><br>
  <input id="two" type="radio" name="items" value="two">
  <label for="two">Choice Two</label><br>
  <input id="three" type="radio" name="items" value="three">
  <label for="three">Choice Three</label>
 </fieldset>
</form>
```

Camper Cat wants information about the ninja level of his users when they sign up for his email list. He's added a set of radio buttons and learned from our last lesson to use label tags with for attributes for each choice. Go Camper Cat! However, his code still needs some help. Change the div tag surrounding the radio buttons to a fieldset tag, and change the p tag inside it to a legend.

==========

Applied Accessibility: Add an Accessible Date Picker

Forms often include the input field, which can be used to create several different form controls. The type attribute on this element indicates what kind of input will be created.

You may have noticed the text and submit input types in prior challenges, and HTML5 introduced an option to specify a date field. Depending on browser support, a date picker shows up in the input field when it's in focus, which makes filling in a form easier for all users.

For older browsers, the type will default to text, so it helps to show users the expected date format in the label or as placeholder text just in case.

Here's an example:

```
<label for="input1">Enter a date:</label>
<input type="date" id="input1" name="input1">
```

==========

Applied Accessibility: Standardize Times with the HTML5 datetime Attribute

Continuing with the date theme, HTML5 also introduced the time element along with a datetime attribute to standardize times. This is an inline element that can wrap a date or time on a page. A valid

format of that date is held by the datetime attribute. This is the value accessed by assistive devices. It helps avoid confusion by stating a standardized version of a time, even if it's written in an informal or colloquial manner in the text.

Here's an example:

Master Camper Cat officiated the cage match between Goro and Scorpion <time datetime="2013-02-13">last Wednesday</time>, which ended in a draw.

Camper Cat's Mortal Kombat survey results are in! Wrap a time tag around the text "Thursday, September 15th" and add a datetime attribute to it set to "2016-09-15".

==========

Applied Accessibility: Make Links Navigable with HTML Access Keys

HTML offers the accesskey attribute to specify a shortcut key to activate or bring focus to an element. This can make navigation more efficient for keyboard-only users.

HTML5 allows this attribute to be used on any element, but it's particularly useful when it's used with interactive ones. This includes links, buttons, and form controls.

Here's an example:

<button accesskey="b">Important Button</button>

==========

Applied Accessibility: Improve Form Field Accessibility with the label ElementPassed

Improving accessibility with semantic HTML markup applies to using both appropriate tag names as well as attributes. The next several challenges cover some important scenarios using attributes in forms.

The label tag wraps the text for a specific form control item, usually the name or label for a choice. This ties meaning to the item and makes the form more readable. The for attribute on a label tag explicitly associates that label with the form control and is used by screen readers.

You learned about radio buttons and their labels in a lesson in the Basic HTML section. In that lesson, we wrapped the radio button input element inside a label element along with the label text in order to make the text clickable. Another way to achieve this is by using the for attribute as explained in this lesson.

The value of the for attribute must be the same as the value of the id attribute of the form control. Here's an example:

```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
</form>
```

==========

Applied Accessibility: Use tabindex to Add Keyboard Focus to an Element

The HTML tabindex attribute has three distinct functions relating to an element's keyboard focus. When it's on a tag, it indicates that element can be focused on. The value (an integer that's positive, negative, or zero) determines the behavior.

Certain elements, such as links and form controls, automatically receive keyboard focus when a user tabs through a page. It's in the same order as the elements come in the HTML source markup. This same functionality can be given to other elements, such as div, span, and p, by placing a tabindex="0" attribute on them. Here's an example:

```
<div tabindex="0">I need keyboard focus!</div>
```

Note: A negative tabindex value (typically -1) indicates that an element is focusable, but is not reachable by the keyboard. This method is generally used to bring focus to content programmatically (like when a div used for a pop-up window is activated), and is beyond the scope of these challenges.