LEARN HTML BY BUILDING A CAT PHOTO APP

Introduction

HTML tags give a webpage its structure. You can use HTML tags to add photos, buttons, and other elements to your webpage.

In this course, you'll learn the most common HTML tags by building your own cat photo app.

Step 1:

HTML elements have an opening and closing tag with content in between.

Here is the basic syntax:

Example Code:

```
<openingTag>content</closingTag>
```

The first element you will learn about is the `h1` element. The `h1` element is a heading element and is used for the main heading of a webpage.

Example Code:

```
<h1>This is a main heading</h1>
```

Change the text of the `h1` element below from `Hello World` to `CatPhotoApp` and watch the change in the browser preview. When you are done, press the "Check Your Code" button to see if it's correct.

Step 2:

The h1 through h6 heading elements are used to signify the importance of content below them. The lower the number, the higher the importance, so h2 elements have less importance than h1 elements.

Example Code:

```
<h1>most important heading element</h1>
<h2>second most important heading element</h2>
<h3>third most important heading element</h3>
<h4>fourth most important heading element</h4>
<h5>fifth most important heading element</h5>
<h6>least important heading element</h6>
```

Only use one h1 element per page and place lower importance headings below higher importance headings.

Below the h1 element, add an h2 element with this text: Cat Photos

Step 3:

The p element is used to create a paragraph of text on websites. Create a p element below your h2 element and give it the following text: See more cat photos in our gallery.

Step 4:

Commenting allows you to leave messages without affecting the browser display. It also allows you to make code inactive. A comment in HTML starts with <!--, contains any number of lines of text, and ends with -->.

Here is an example of a comment with the TODO: Remove h1:

Example Code

<!-- TODO: Remove h1 -->

Add a comment above the p element with this text: TODO: Add link to cat photos

Step 5:

HTML5 has some elements that identify different content areas. These elements make your HTML easier to read and help with Search Engine Optimization (SEO) and accessibility.

The main element is used to represent the main content of the body of an HTML document. Content inside the main element should be unique to the document and should not be repeated in other parts of the document.

Example Code

```
<main>

  <h1>Most important content of the document</h1>

  <p>Some more important content...</p>

</main>
```

Identify the main section of this page by adding a <main> opening tag before the h1 element, and a </main> closing tag after the p element.

Step 6:

In the previous step, you put the h1, h2, comment, and p elements inside the main element. This is called *nesting*. Nested elements should be placed two spaces further to the right of the element they are nested in. This spacing is called indentation and it is used to make HTML easier to read.

Here is an example of nesting and indentation:

Example Code

```
<main>

  <h1>Most important content of the document</h1>

  <p>Some more important content...</p>

</main>
```

The h1 element, h2 element and the comment are indented two spaces more than the main element in the code below. Use the space bar on your keyboard to add two more spaces in front of the p element so that it is indented properly as well.

Step 7:

You can add images to your website by using the img element. img elements have an opening tag without a closing tag. An element without a closing tag is known as a void element.

Add an img element below the p element. At this point, no image will show up in the browser.

Step 8:

HTML attributes are special words used inside the opening tag of an element to control the element's behavior. The src attribute in an img element specifies the image's URL (where the image is located).

Here is an example of an img element with a src attribute pointing to the freeCodeCamp logo:

Example Code:

```
<img
src="https://cdn.freecodecamp.org/platform/universal/fcc_secondary.svg
">
```

Inside the existing img element, add a src attribute with this URL:

https://cdn.freecodecamp.org/curriculum/cat-photo-app/relaxing-cat.jpg

Step 9:

All img elements should have an alt attribute. The alt attribute's text is used for screen readers to improve accessibility and is displayed if the image fails to load.

Here is an example of an img element with an alt attribute:

Example Code:

```
<img src="cat.jpg" alt="A cat">
```

Inside the img element, add an alt attribute with this text: A cute orange cat lying on its back

Step 10:

You can link to another page with the anchor (a) element.

Here is an example linking to https://www.freecodecamp.org:

Example Code:

```
<a href="https://www.freecodecamp.org"></a>
```

Add an anchor element after the paragraph that links to https://freecatphotoapp.com. At this point, the link won’t show up in the preview.

Step 11:

A link's text must be placed between the opening and closing tags of an anchor (`a`) element.

Here is an example of a link with the text `click here to go to freeCodeCamp.org`:

Example Code:

```
<a href="https://www.freecodecamp.org">click here to go to
freeCodeCamp.org</a>
```

Add the anchor text `link to cat pictures` to the anchor element. This will become the link's text.

Step 12:

You can turn any text into a link, such as the text inside of a `p` element.

Example Code

```
<p>I think <a href="https://www.freecodecamp.org">freeCodeCamp</a> is
great.</p>
```

In the text of your `p` element, turn the words `cat photos` into a link by adding opening and closing anchor (`a`) tags around these words. Then set the `href` attribute to https://freecatphotoapp.com

Step 13:

Now that you turned the text `cat photos` inside the `p` element into a link, you don't need the second link below the `p` element. Delete the entire anchor element below the `p` element.

Step 14:

To open links in a new tab, you can use the `target` attribute on the anchor (`a`) element.

The target attribute specifies where to open the linked document. target="_blank" opens the linked document in a new tab or window.

Here is the basic syntax for an a element with a target attribute:

Example Code:

```
<a href="https://www.freecodecamp.org"
target="_blank">freeCodeCamp</a>
```

Add a target attribute with the value _blank to the anchor (a) element's opening tag, so that the link opens in a new tab.

Step 15:

In previous steps you used an anchor element to turn text into a link. Other types of content can also be turned into a link by wrapping it in anchor tags.

Here is an example of turning an image into a link:

Example Code:

```
<a href="example-link">

  <img src="image-link.jpg" alt="A photo of a cat.">

</a>
```

Turn the image into a link by surrounding it with necessary element tags. Use https://freecatphotoapp.com as the anchor's href attribute value.

Step 16:

Before adding any new content, you should make use of a section element to separate the cat photos content from the future content.

The section element is used to define sections in a document, such as chapters, headers, footers, or any other sections of the document. It is a semantic element that helps with SEO and accessibility.

Example Code:

```
<section>

  <h2>Section Title</h2>

  <p>Section content...</p>

</section>
```

Take your h2, comment, p, and anchor (a) elements and nest them in a section element.

Step 17:

It is time to add a new section. Add a second section element below the existing section element.

Step 18:

Within the second section element, add a new h2 element with the text Cat Lists.

Step 19:

When you add a lower rank heading element to the page, it's implied that you're starting a new subsection.

After the last h2 element of the second section element, add an h3 element with this text: Things cats love:

Step 20:

To create an unordered list of items, you can use the ul element.

After the h3 element with the Things cats love: text, add an unordered list (ul) element. Note that nothing will be displayed at this point.

Step 21:

The li element is used to create a list item in an ordered or unordered list.

Here is an example of list items in an unordered list:

Example Code:

```
<ul>

  <li>milk</li>

  <li>cheese</li>

</ul>
```

Within the ul element nest three list items to display three things cats love: cat nip, laser pointers, lasagna

Step 22:

After the unordered list, add a new image with a src attribute value set to:

https://cdn.freecodecamp.org/curriculum/cat-photo-app/lasagna.jpg

And its alt attribute value to: A slice of lasagna on a plate.

Step 23:

The figure element represents self-contained content and will allow you to associate an image with a caption.

Nest the image you just added within a figure element.

Step 24:

A figure caption (figcaption) element is used to add a caption to describe the image contained within the figure element.

Here is an example of a figcaption element with the caption of A cute cat:

Example Code:

```
<figure>

  <img src="image.jpg" alt="A description of the image">

  <figcaption>A cute cat</figcaption>

</figure>
```

After the image nested in the figure element, add a figcaption element with text set to: Cats love lasagna.

Step 25:

To place emphasis on a specific word or phrase, you can use the em element.

Emphasize the word love in the figcaption element by wrapping it in an emphasis em element.

Step 26:

After the figure element, add another h3 element with the text: Top 3 things cats hate:

Step 27:

The code for an ordered list (ol) is similar to an unordered list, but list items in an ordered list are numbered when displayed.

After the second section element's last h3 element, add an ordered list with these three list items: flea treatment, thunder, other cats

Step 28:

After the ordered list, add another figure element.

Step 29:

Inside the figure element you just added, nest an img element with a src attribute set to https://cdn.freecodecamp.org/curriculum/cat-photo-app/cats.jpg.

Step 30:

To improve accessibility of the image you added, add an alt attribute with the text: Five cats looking around a field.

Step 31:

After the last img element, add a figcaption element with the text Cats hate other cats.

Step 32:

The strong element is used to indicate that some text is of strong importance or urgent.

In the figcaption you just added, indicate that hate is of strong importance by wrapping it in a strong element.

Step 33:

It is time to add a new section. Add a third section element below the second section element.

Step 34:

Inside the third section element, add an h2 element with the text: Cat Form

Step 35:

Now you will add a web form to collect information from users.

The form element is used to get information from a user like their name, email, and other details.

After the Cat Form heading, add a form element.

Step 36:

The action attribute indicates where form data should be sent.

Here is an example of a form element with an action attribute:

Example Code:

```
<form action="/submit-url"></form>
```

In the example, `action="/submit-url"` tells the browser that the form data should be sent to the path `/submit-url`.

Add an `action` attribute with the value `https://freecatphotoapp.com/submit-cat-photo` to the `form` element.

Step 37:

The `input` element allows you several ways to collect data from a web form. Like `img` elements, `input` elements are a void element and do not need closing tags.

Nest an `input` element in the `form` element.

Step 38:

There are many kinds of inputs you can create using the `type` attribute. You can easily create a password field, reset button, or a control to let users select a file from their computer.

Create a text field to get text input from a user by adding the `type` attribute with the value `text` to the `input` element.

Step 39:

In order for a form's data to be accessed by the location specified in the `action` attribute, you must give the text field a `name` attribute and assign it a value to represent the data being submitted.

Here is an example of an `input` element with a `name` attribute:

Example Code:

```
<input type="text" name="name">
```

Add the `name` attribute with the value `catphotourl` to your text field.

Step 40:

Placeholder text is used to give people a hint about what kind of information to enter into an input.

Here is an example of an `input` element with a placeholder set to `Ex. Jane Doe`:

Example Code:

```
<input type="text" placeholder="Ex. Jane Doe">
```

Add the placeholder text `cat photo URL` to your `input` element.

Step 41:

To prevent a user from submitting your form when required information is missing, you need to add the `required` attribute to an `input` element. There's no need to set a value to the `required` attribute. Instead, just add the word `required` to the `input` element, making sure there is space between it and other attributes.

Step 42:

The `button` element is used to create a clickable button.

Add a `button` element with the text `Submit` below the `input` element. The default behavior of clicking a form button without any attributes submits the form to the location specified in the form's `action` attribute.

Step 43:

Even though you added your button below the text input, they appear next to each other on the page. That's because both `input` and `button` elements are inline elements, which don't appear on new lines.

The button you added will submit the form by default. However, relying on default behavior may cause confusion. Add the `type` attribute with the value `submit` to the `button` to make it clear that it is a submit button.

Step 44:

You can use radio buttons for questions where you want only one answer out of multiple options.

Here is an example of a radio button with the option of `cat`:

Example Code:

```
<input type="radio"> cat
```

Remember that an `input` element is a void element.

Before the text input, add a radio button with the option set as: `Indoor`

Step 45:

`label` elements are used to help associate the text for an `input` element with the `input` element itself (especially for assistive technologies like screen readers).

Here is an example of a `label` element with a `radio` button:

Example Code:

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

In the example, clicking on the word `"cat"` will also select the `radio` button.

Nest your `radio` button inside a `label` element.

Step 46:

The `id` attribute is used to identify specific HTML elements. Each `id` attribute's value must be unique from all other `id` values for the entire page.

Here is an example of an `input` element with an `id` attribute:

Example Code:

```
<input id="email">
```

Add an `id` attribute with the value `indoor` to the radio button. When elements have multiple attributes, the order of the attributes doesn't matter.

Step 47:

Create another radio button below the first one. Nest it inside a `label` element with `Outdoor` as the `label` text. Give the radio button an `id` attribute with `outdoor` as the value.

Step 48:

Notice that both radio buttons can be selected at the same time. To make it so selecting one radio button automatically deselects the other, both buttons must have a `name` attribute with the same value.

Here is an example of two radio buttons with the same `name` attribute:

Example Code:

```
<input type="radio" name="meal"> Breakfast
```

```
<input type="radio" name="meal"> Lunch
```

Add the name attribute with the value indoor-outdoor to both radio buttons.

Step 49:

If you select the Indoor radio button and submit the form, the form data for the button is based on its name and value attributes. Since your radio buttons do not have a value attribute, the form data will include indoor-outdoor=on, which is not useful when you have multiple buttons.

Add a value attribute to both radio buttons. For convenience, set the button's value attribute to the same value as its id attribute.

Step 50:

The fieldset element is used to group related inputs and labels together in a web form. fieldset elements are block-level elements, meaning that they appear on a new line.

Nest the Indoor and Outdoor radio buttons within a fieldset element, and don't forget to indent the radio buttons.

Step 51:

The legend element acts as a caption for the content in the fieldset element. It gives users context about what they should enter into that part of the form.

Add a legend element with the text Is your cat an indoor or outdoor cat? above both of the radio buttons.

Step 52:

Next, you are going to add some new form input elements, so add another fieldset element directly below the current fieldset element.

Step 53:

Add a legend element with the text What's your cat's personality? inside the second fieldset element.

Step 54:

Forms commonly use checkboxes for questions that may have more than one answer. The input element with a type attribute set to checkbox creates a checkbox.

Under the legend element you just added, add an input with its type attribute set to checkbox and give it the option of: Loving

Step 55:

Add an id attribute with the value loving to the checkbox input.

Step 56:

There's another way to associate an input element's text with the element itself. You can nest the text within a label element and add a for attribute with the same value as the input element's id attribute.

Given an input element as below:

Example Code:

```
<input id="breakfast" type="radio" name="meal" value="breakfast">
```

An example of a `label` element that is associated to this `input` element is:

Example Code:

```
<label for="breakfast">Breakfast</label>
```

Associate the text `Loving` with the checkbox by nesting only the text `Loving` in a `label` element and giving it an appropriate `for` attribute.

Step 57:

Add the `name` attribute with the value `personality` to the checkbox `input` element.

While you won't notice this in the browser, doing this makes it easier for a server to process your web form, especially when there are multiple checkboxes.

Step 58:

Add another checkbox after the one you just added. The `id` attribute value should be `lazy` and the `name` attribute value should be the same as the last checkbox.

Also add a `label` element to the right of the new checkbox with the text `Lazy`. Make sure to associate the `label` element with the new checkbox using the `for` attribute.

Step 59:

Add a final checkbox after the previous one with an `id` attribute value of `energetic`. The `name` attribute should be the same as the previous checkbox.

Also add a `label` element to the right of the new checkbox with text `Energetic`. Make sure to associate the `label` element with the new checkbox.

Step 60:

Like radio buttons, form data for selected checkboxes are `name` / `value` attribute pairs. While the `value` attribute is optional, it's best practice to include it with any checkboxes or radio buttons on the page.

Add a `value` attribute to each checkbox. For convenience, set each checkbox's `value` attribute to the same value as its `id` attribute.

Step 61:

In order to make a checkbox checked or radio button selected by default, you need to add the `checked` attribute to it.

Here is an example of a radio button with the `checked` attribute:

Example Code:

```
<input checked type="radio" name="meal" value="breakfast"> Breakfast
```

There's no need to set a value to the `checked` attribute. Instead, just add the word `checked` to the `input` element, making sure there is space between it and other attributes.

Make the first radio button and the first checkbox selected by default.

Step 62:

The `footer` element is used to define a footer for a document or section. A footer typically contains information about the author of the document, copyright data, links to terms of use, contact information, and more.

After the `main` element, add a `footer` element.

Step 63:

Nest a `p` element with the text `No Copyright - freeCodeCamp.org` within the `footer` element.

Step 64:

Turn the existing `freeCodeCamp.org` text into a link by enclosing it in an anchor (`a`) element. The `href` attribute should be set to https://www.freecodecamp.org.

Step 65:

Notice that everything you've added to the page so far is inside the `body` element. All page content elements that should be rendered to the page go inside the `body` element. However, other important information goes inside the `head` element.

The `head` element is used to contain metadata about the document, such as its title, links to stylesheets, and scripts. Metadata is information about the page that isn't displayed directly on the page.

Add a `head` element above the `body` element.

Step 66:

The `title` element determines what browsers show in the title bar or tab for the page.

Add a `title` element within the `head` element using the text below: `CatPhotoApp`

Step 67:

Notice that the entire contents of the page are nested within an `html` element. The `html` element is the root element of an HTML page and wraps all content on the page.

You can also specify the language of your page by adding the `lang` attribute to the `html` element.

Add the `lang` attribute with the value `en` to the opening `html` tag to specify that the language of the page is English.

Step 68:

All pages should begin with `<!DOCTYPE html>`. This special string is known as a declaration and ensures the browser tries to meet industry-wide specifications.

`<!DOCTYPE html>` tells browsers that the document is an HTML5 document which is the latest version of HTML.

Add this declaration as the first line of the code.

Step 69:

You can set browser behavior by adding `meta` elements in the `head`. Here's an example:

Example Code:

```
<meta attribute="value">
```

Inside the `head` element, nest a `meta` element with an attribute named `charset`. Set the value to `utf-8` which tells the browser how to encode characters for the page.

Note that the `meta` element is a void element.

With that last change, you have completed the Cat Photo App project. Congratulations!