codecademy

Elements and Structure

`<a>` Anchor Element

The `<a>` anchor element is used to create hyperlinks in an HTML document. The hyperlinks can point to other webpages, files on the same server, a location on the same page, or any other URL via the hyperlink reference attribute, `href`. The `href` determines the location the anchor element points to.

```
<!-- Creating text links -->
<a href="http://www.codecademy.com">Visit
this site</a>

<!-- Creating image links -->
<a href="http://www.codecademy.com">
        <img src="logo.jpg">Click this
image
</a>
```

`<head>` Head Element

The `<head>` element contains general information about an HTML page that isn't displayed on the page itself. This information is called metadata and includes things like the title of the HTML document and links to stylesheets.

```
<!DOCTYPE html>
<html>
  <head>
    <!-- Metadata is contained in this
element-->
  </head>
</html>
```

`<target>` Target Attribute

The `target` attribute on an `<a>` anchor element specifies where a hyperlink should be opened. A `target` value of `"_blank"` will tell the browser to open the hyperlink in a new tab in modern browsers, or in a new window in older browsers or if the browser has had settings changed to open hyperlinks in a new window.

```
<a href="https://www.google.com"
target="_blank">This anchor element links
to google and will open in a new tab or
window.</a>
```

Indentation

HTML code should be formatted such that the indentation level of text increases once for each level of nesting.
It is a common convention to use two or four `space` per level of nesting.

```
<div>
  <h1>Heading</h1>

  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</div>
```

Link to a Different Part of the Page `#`

The anchor element `<a>` can create hyperlinks to different parts of the same HTML document using the `href` attribute to point to the desired location with `#` followed by the `id` of the element to link to.

```
<div>
  <p id="id-of-element-to-link-to">A
different part of the page!</p>
</div>

<a href="#id-of-element-to-link-to">Take
me to a different part of the page</a>
```

`<html>` HTML Element

The `<html>` element, the root of an HTML document, should be added after the `!DOCTYPE` declaration. All content/structure for an HTML document should be contained between the opening and closing `<html>` tags.

```
<!DOCTYPE html>
<html>
  <!-- I'm a comment -->
</html>
```

Comments

In HTML, comments can be added between an opening `<!--` and closing `-->`. Content inside of comments will not be rendered by browsers, and are usually used to describe a part of code or provide other details. Comments can span single or multiple lines.

```
<!-- Main site content -->
<div>Content</div>

<!--
  Comments can be
  multiple lines long.
-->
```

Whitespace

Whitespace, such as line breaks, added to an HTML document between block-level elements will generally be ignored by the browser and are not added to increase spacing on the rendered HTML page. Rather, whitespace is added for organization and easier reading of the HTML document itself.

```
<p>Test paragraph</p>

<!-- The whitespace created by this line, and above/below this line is ignored by the browser-->

<p>Another test paragraph, this will sit right under the first paragraph, no extra space between.</p>
```

`<title>` Title Element

The `<title>` element contains a text that defines the title of an HTML document. The title is displayed in the browser's title bar or tab in which the HTML page is displayed. The `<title>` element can only be contained inside a document's `<head>` element.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the HTML page</title>
  </head>
</html>
```

File Path

URL paths in HTML can be absolute paths, like a full URL, for example: `https://developer.mozilla.org/en-US/docs/Learn` or a relative file path that links to a local file in the same folder or on the same server, for example: `./style.css`. Relative file paths begin with `./` followed by a path to the local file. `./` tells the browser to look for the file path from the current folder.

```
<a
href="https://developer.mozilla.org/en-US/docs/Web">The URL for this anchor element is an absolute file path.</a>

<a href="./about.html">The URL for this anchor element is a relative file path.
</a>
```

Document Type Declaration

The document type declaration `<!DOCTYPE html>` is required as the first line of an HTML document. The doctype declaration is an instruction to the browser about what type of document to expect and which version of HTML is being used, in this case it's HTML5.

```
<!DOCTYPE html>
```

HTML

HTML (HyperText Markup Language) is used to give content to a web page and instructs web browsers on how to structure that content.

Element Content

The content of an HTML element is the information between the opening and closing tags of an element.

```
<h1>Codecademy is awesome! 🙂</h1>
```

`<li>` List Item Element

The `<li>` list item element create list items inside:

- Ordered lists `<ol>`
- Unordered lists `<ul>`

```
<ol>
  <li>Head east on Prince St</li>
  <li>Turn left on Elizabeth</li>
</ol>

<ul>
  <li>Cookies</li>
  <li>Milk</li>
</ul>
```

`<video>` Video Element

The `<video>` element embeds a media player for video playback. The `src` attribute will contain the URL to the video. Adding the `controls` attribute will display video controls in the media player.
Note: The content inside the opening and closing tag is shown as a fallback in browsers that don't support the element.

```
<video src="test-video.mp4" controls>
  Video not supported
</video>
```

`<em>` Emphasis Element

The `<em>` element emphasizes text and browsers will usually *italicize* the emphasized text by default.

```
<p>This <em>word</em> will be emphasized
in italics.</p>
```

`<ol>` Ordered List Element

The `<ol>` ordered list element creates a list of items in sequential order. Each list item appears numbered by default.

```
<ol>
  <li>Preheat oven to 325 F 👸</li>
  <li>Drop cookie dough 🍪</li>
  <li>Bake for 15 min ⏰</li>
</ol>
```

`<div>` Div Element

The `<div>` element is used as a container that divides an HTML document into sections and is short for "division". `<div>` elements can contain *flow content* such as headings, paragraphs, links, images, etc.

```
<div>
  <h1>A section of grouped elements</h1>
  <p>Here's some text for the section</p>
</div>
<div>
  <h1>Second section of grouped
elements</h1>
  <p>Here's some text</p>
</div>
```

HTML Structure

HTML is organized into a family tree structure. HTML elements can have parents, grandparents, siblings, children, grandchildren, etc.

```
<body>
  <div>
    <h1>It's div's child and body's grandchild</h1>
    <h2>It's h1's sibling</h2>
  </div>
</body>
```

Closing Tag

An HTML closing tag is used to denote the end of an HTML element. The syntax for a closing tag is a left angle bracket `<` followed by a forward slash `/` then the element name and a right angle bracket to close `>`.

```
<body>
  ...
</body>
```

Attribute Name and Values

HTML attributes consist of a name and a value using the following syntax: `name="value"` and can be added to the opening tag of an HTML element to configure or change the behavior of the element.

```
<elementName name="value"></elementName>
```

`<br>` Line Break Element

The `<br>` line break element will create a line break in text and is especially useful where a division of text is required, like in a postal address. The line break element requires only an opening tag and must not have a closing tag.

```
A line break haiku.<br>
Poems are a great use case.<br>
Oh joy! A line break.
```

`<img>` Image Element

HTML image `<img>` elements embed images in documents. The `src` attribute contains the image URL and is mandatory. `<img>` is an *empty element* meaning it should not have a closing tag.

```
<img src="image.png">
```

`<h1>` - `<h6>` Heading Elements

HTML can use six different levels of heading elements. The heading elements are ordered from the highest level `<h1>` to the lowest level `<h6>`.

```
<h1>Breaking News</h1>
<h2>This is the 1st subheading</h2>
<h3>This is the 2nd subheading</h3>
...
<h6>This is the 5th subheading</h6>
```

`<p>` Paragraph Element

The `<p>` paragraph element contains and displays a block of text.

```
<p>This is a block of text! Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>
```

Unique ID Attributes

In HTML, specific and unique `id` attributes can be assigned to different elements in order to differentiate between them.
When needed, the `id` value can be called upon by CSS and JavaScript to manipulate, format, and perform specific instructions on that element and that element only. Valid `id` attributes should begin with a letter and should only contain letters (`a-Z`), digits (`0-9`), hyphens (`-`), underscores (`_`), and periods (`.`).

```
<h1 id="A1">Hello World</h1>
```

HTML Attributes

HTML attributes are values added to the opening tag of an element to configure the element or change the element's default behavior. In the provided example, we are giving the `<p>` (paragraph) element a unique identifier using the `id` attribute and changing the color of the default text using the `style` attribute.

```
<p id="my-paragraph" style="color: green;">Here's some text for a paragraph that is being altered by HTML attributes</p>
```

`<ul>` Unordered List Element

The `<ul>` unordered list element is used to create a list of items in no particular order. Each individual list item will have a bullet point by default.

```
<ul>
  <li>Play more music 🎸</li>
  <li>Read more books 📚</li>
</ul>
```

`alt` Attribute

An `<img>` element can have alternative text via the `alt` attribute. The alternative text will be displayed if an image fails to render due to an incorrect URL, if the image format is not supported by the browser, if the image is blocked from being displayed, or if the image has not been received from the URL.
The text will be read aloud if screen reading software is used and helps support visually impaired users by providing a text descriptor for the image content on a webpage.

```
<img src="path/to/image" alt="text describing image" />
```

`<body>` Body Element

The `<body>` element represents the content of an HTML document. Content inside `<body>` tags are rendered on the web browsers.
Note: There can be only one `<body>` element in a document.

```
<body>
  <h1>Learn to code with Codecademy :)</h1>
</body>
```

`<span>` Span Element

The `<span>` element is an inline container for text and can be used to group text for styling purposes. However, as `<span>` is a generic container to separate pieces of text from a larger body of text, its use should be avoided if a more semantic element is available.

```
<p><span>This text</span> may be styled differently than the surrounding text.</p>
```

`<strong>` Strong Element

The `<strong>` element highlights important, serious, or urgent text and browsers will normally render this highlighted text in **bold** by default.

```
<p>This is <strong>important</strong> text!</p>
```

HTML Element

An HTML element is a piece of content in an HTML document and uses the following syntax: opening tag + content + closing tag. In the code provided:

- `<p>` is the opening tag.
- `Hello World!` is the content.
- `</p>` is the closing tag.

```
<p>Hello World!</p>
```

HTML Tag

The syntax for a single HTML tag is an opening angle bracket `<` followed by the element name and a closing angle bracket `>`. Here is an example of an opening `<div>` tag.

```
<div>
```

Print Share

Tables

`<tr>` Table Row Element

The table row element, `<tr>`, is used to add rows to a table before adding table data and table headings.

```
<table>
  <tr>
    ...
  </tr>
</table>
```

`<td>` Table Data Element

The table data element, `<td>`, can be nested inside a table row element, `<tr>`, to add a cell of data to a table.

```
<table>
  <tr>
    <td>cell one data</td>
    <td>cell two data</td>
  </tr>
</table>
```

`<thead>` Table Head Element

The table head element, `<thead>`, defines the headings of table columns encapsulated in table rows.

```
<table>
 <thead>
   <tr>
     <th>heading 1</th>
     <th>heading 2</th>
   </tr>
 </thead>
 <tbody>
   <tr>
     <td>col 1</td>
     <td>col 2</td>
   </tr>
 </tbody>
</table>
```

`rowspan` Attribute

Similar to `colspan`, the `rowspan` attribute on a table header or table data element indicates how many rows that particular cell should span within the table. The `rowspan` value is set to 1 by default and will take any positive integer up to 65534.

```
<table>
  <tr>
    <th>row 1:</th>
    <td>col 1</td>
    <td>col 2</td>
  </tr>
  <tr>
    <th rowspan="2">row 2 (this row will span 2 rows):</th>
    <td>col 1</td>
    <td>col 2</td>
  </tr>
  <tr>
    <td>col 1</td>
    <td>col 2</td>
  </tr>
   <tr>
    <th>row 3:</th>
    <td>col 1</td>
    <td>col 2</td>
  </tr>
</table>
```

`<tbody>` Table Body Element

The table body element, `<tbody>`, is a semantic element that will contain all table data other than table heading and table footer content. If used, `<tbody>` will contain all table row `<tr>` elements, and indicates that `<tr>` elements make up the body of the table. `<table>` cannot have both `<tbody>` and `<tr>` as direct children.

```
<table>
 <tbody>
   <tr>
     <td>row 1</td>
   </tr>
   <tr>
     <td>row 2</td>
   </tr>
   <tr>
     <td>row 3</td>
   </tr>
 </tbody>
</table>
```

`<th>` Table Heading Element

The table heading element, `<th>`, is used to add titles to rows and columns of a table and must be enclosed in a table row element, `<tr>`.

```
<table>
  <tr>
    <th>column one</th>
    <th>column two</th>
  </tr>
  <tr>
    <td>1</td>
    <td>2</td>
  </tr>
</table>
```

`colspan` Attribute

The `colspan` attribute on a table header `<th>` or table data `<td>` element indicates how many columns that particular cell should span within the table. The `colspan` value is set to 1 by default and will take any positive integer between 1 and 1000.

```
<table>
  <tr>
    <th>row 1:</th>
    <td>col 1</td>
    <td>col 2</td>
    <td>col 3</td>
  </tr>
  <tr>
    <th>row 2:</th>
    <td colspan="2">col 1 (will span 2
columns)</td>
    <td>col 2</td>
    <td>col 3</td>
  </tr>
</table>
```

`<tfoot>` Table Footer Element

The table footer element, `<tfoot>`, uses table rows to give footer content or to summarize content at the end of a table.

```
<table>
 <thead>
   <tr>
     <th>heading 1</th>
     <th>heading 2</th>
   </tr>
 </thead>
 <tbody>
   <tr>
     <td>col 1</td>
     <td>col 2</td>
   </tr>
 </tbody>
 <tfoot>
   <tr>
     <td>summary of col 1</td>
     <td>summary of col 2</td>
   </tr>
 </tfoot>
</table>
```

`<table>` Table Element

In HTML, the `<table>` element has content that is used to represent a two-dimensional table made of rows and columns.

```
<table>
  <!-- rows and columns will go here -->
</table>
```

Forms

`<input>`: Checkbox Type

When using an HTML `input` element, the `type="checkbox"` attribute will render a single checkbox item. To create a group of checkboxes related to the same topic, they should all use the same `name` attribute. Since it's a checkbox, multiple checkboxes can be selected for the same topic.

```
<input type="checkbox" name="breakfast"
value="bacon">Bacon <br>
<input type="checkbox" name="breakfast"
value="eggs">Eggs <br>
<input type="checkbox" name="breakfast"
value="pancakes">Pancakes <br>
```

`<textarea>` Element

The `textarea` element is used when creating a text-box for multi-line input (e.g. a comment section). The element supports the `rows` and `cols` attributes which determine the height and width, respectively, of the element.

When rendered by the browser, `textarea` fields can be stretched/shrunk in size by the user, but the `rows` and `cols` attributes determine the initial size.

Unlike the `input` element, the `<textarea>` element has both opening and closing tags. The `value` of the element is the content in between these tags (much like a `<p>` element). The code block shows a `<textarea>` of size 10x30 and with a `name` of `"comment"`.

```
<textarea rows="10" cols="30"
name="comment"></textarea>
```

`<form>` Element

The HTML `<form>` element is used to collect and send information to an external source.
`<form>` can contain various input elements. When a user submits the form, information in these input elements is passed to the source which is named in the `action` attribute of the form.

```
<form method="post"
action="http://server1">
  Enter your name:
  <input type="text" name="fname">
  <br/>
  Enter your age:
  <input type="text" name="age">
  <br/>
  <input type="submit" value="Submit">
</form>
```

`<input>`: Number Type

HTML input elements can be of type `number`. These input fields allow the user to enter only numbers and a few special characters inside the field.
The example code block shows an input with a type of `number` and a name of `balance`. When the input field is a part of a form, the form will receive a key-value pair with the format: `name: value` after form submission.

```
<input type="number" name="balance" />
```

`<input>` Element

The HTML `<input>` element is used to render a variety of input fields on a webpage including text fields, checkboxes, buttons, etc. `<input>` element have a `type` attribute that determines how it gets rendered to a page.
The example code block will create a text input field and a checkbox input field on a webpage.

```
<label for="fname">First name:</label>
<input type="text" name="fname"
id="fname"><br>

<input type="checkbox" name="vehicle"
value="Bike"> I own a bike
```

`<input>`: Range Type

A slider can be created by using the `type="range"` attribute on an HTML `input` element. The range slider will act as a selector between a minimum and a maximum value. These values are set using the `min` and `max` attributes respectively. The slider can be adjusted to move in different steps or increments using the `step` attribute.
The range slider is meant to act more as a visual widget to adjust between 2 values, where the relative position is important, but the precise value is not as important. An example of this can be adjusting the volume level of an application.

```
<input type="range" name="movie-rating"
min="0" max="10" step="0.1">
```

`<select>` Element

The HTML `<select>` element can be used to create a dropdown list. A list of choices for the dropdown list can be created using one or more `<option>` elements. By default, only one `<option>` can be selected at a time.
The value of the selected `<select>`'s `name` and the `<option>`'s `value` attribute are sent as a key-value pair when the form is submitted.

```
<select name="rental-option">
  <option value="small">Small</option>
  <option value="family">Family
Sedan</option>
  <option value="lux">Luxury</option>
</select>
```

Submitting a Form

Once we have collected information in a form we can send that information somewhere else by using the `action` and `method` attribute. The `action` attribute tells the form to send the information. A URL is assigned that determines the recipient of the information. The `method` attribute tells the form what to do with that information once it's sent. An HTTP verb is assigned to the `method` attribute that determines the action to be performed.

```
<form action="/index3.html" method="PUT">
</form>
```

`<input>`: Text Type

HTML `<input>` elements can support text input by setting the attribute `type="text"`. This renders a single row input field that users can type text inside.
The value of the `<input>`'s `name` and `value` attribute of the element are sent as a key-value pair when the form is submitted.

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

`<datalist>` Element

When using an HTML input, a basic search/autocomplete functionality can be achieved by pairing an `<input>` with a `<datalist>`. To pair a `<input>` with a `<datalist>` the `<input>`'s `list` value must match the value of the `id` of the `<datalist>`. The `datalist` element is used to store a list of `<option>`s.
The list of data is shown as a dropdown on an `input` field when a user clicks on the input field. As the user starts typing, the list will be updated to show elements that best match what has been typed into the input field. The actual list items are specified as multiple `option` elements nested inside the `datalist`.
`datalist`s are ideal when providing users a list of pre-defined options, but to also allow them to write alternative inputs as well.

```
<input list="ide">

<datalist id="ide">
  <option value="Visual Studio Code" />
  <option value="Atom" />
  <option value="Sublime Text" />
</datalist>
```

`<input>`: Radio Button Type

HTML `<input>` elements can be given a `type="radio"` attribute that renders a single radio button. Multiple radio buttons of a related topic are given the same `name` attribute value. Only a single option can be chosen from a group of radio buttons.
The value of the selected/checked `<input>`'s `name` and `value` attribute of this element are sent as a key-value pair when the form is submitted.

```
<input name="delivery_option"
type="radio" value="pickup" />
<input name="delivery_option"
type="radio" value="delivery" />
```

Submittable Input

HTML `<input>` elements can have a type attribute set to submit, by adding `type="submit"`. With this attribute included, a submit button will be rendered and, by default, will submit the `<form>` and execute its action.
The text of a submit button is set to `Submit` by default but can also be changed by modifying the `value` attribute.

`<input>` `name` Attribute

In order for a form to send data, it needs to be able to put it into key-value pairs. This is achieved by setting the `name` attribute of the `input` element. The `name` will become the `key` and the `value` of the input will become the `value` the form submits corresponding to the key.
It's important to remember that the name is not the same as the ID in terms of form submission. The ID and the name of the input may be the same, but the value will only be submitted if the `name` attribute is specified.
In the code example, the first input will be submitted by the form, but the second one will not.

```
<input name="username" id="username" />
<input id="address" />
```

`<label>` Element

The HTML `<label>` element provides identification for a specific `<input>` based on matching values of the `<input>`'s `id` attribute and the `<label>`'s `for` attribute. By default, clicking on the `<label>` will focus the field of the related `<input>`.
The example code will create a text input field with the label text "Password: " next to it. Clicking on "Password: " on the page will focus the field for the related `<input>`.

```
<label for="password ">Password:</label>
<input type="text" id="password"
name="password">
```

`<input>` Password Type

The HTML `<input>` element can have the attribute `type="password"` that renders a single row input field which allows the user to type censored text inside the field. It is used to type in sensitive information.
The value of this `<input>`'s `name` and `value` (actual value and not the censored version) attribute of this element are sent as a key-value pair when the form is submitted.
The code block shows an example of the fields for a basic login form - the username and password fields.

```
<input type="text" name="username" />
<input type="password" name="password" />
```

`required` Attribute

In HTML, input fields have an attribute called `required` which specifies that the field must include a value.
The example code block shows an input field that is required. The attribute can be written as `required="true"` or simply `required`.

```
<input type="password" name="password"
required >
```

`max` Attribute

HTML `<input>` s of type `number` have an attribute called `max` that specifies the maximum value for the input field.
The code block shows an `input` number field that is set to have a maximum value of `20`. Any value larger than `20` will mark the input field as having an error.

```
<input type="number" max="20">
```

`maxlength` Attribute

In HTML, input fields with type `text` have an attribute called `maxlength` that specifies the maximum number of characters that can be entered into the field. The code block shows an input text field that accepts text that has a maximum length of 140 characters.

```
<input type="text" name="tweet"
maxlength="140">
```

`pattern` Attribute

In a `text` input element, the `pattern` attribute uses a regular expression to match against (or validate) the value of the `<input>`, when the form is submitted.

```
<form action="/action_page.php">
  Country code:
  <input type="text" name="country_code"
         pattern="[A-Za-z]{3}"
         title="Three letter country
code">
  <input type="submit">
</form>
```

`minlength` Attribute

In HTML, an input field of type `text` has an attribute that supports minimum length validation. To check that the input text has a minimum length, add the `minlength` attribute with the character count. The example code block shows an example of a text field that has a minimum length of `6`.

```
<input type="text" name="username"
minlength="6" />
```

HTML Form Validators

HTML forms allow you to specify different kinds of validation for your input fields to make sure that data is entered correctly before being submitted. HTML supports a number of different validators, including things like minimum value, minimum/maximum length, etc. The validators are specified as attributes on the `input` field.

`min` Attribute

In HTML, input fields with type `number` have an attribute called `min` that specifies the minimum value that can be entered into the field. The code block provided shows an input number field that accepts a number with minimum value 1.

```
<input type="number" name="rating"
min="1" max="10">
```

Semantic HTML

Semantic HTML

Semantic HTML introduces meaning to the code we write. Before Semantic HTML the elements didn't have any meaning as to what it does or what content goes in it. An element such as `<div>` was used as a general-purpose element to create things from headers to footers to articles. With Semantic HTML we were introduced to elements that tell developers and browsers exactly what it does and what content should go in it.

```
<!--Non Semantic HTML-->
<div id="footer">
 <p>this is a footer</p>
</div>

<!--Semantic HTML-->
<footer>
 <p>this is a footer</p>
</footer>
```

Element Placement

Semantic HTML introduces elements that can tell developers exactly what the element does or where it's placed based on the name of that element. Some of these elements are `<header>`, `<nav>`, `<main>`, and `<footer>`. `<header>` describes the content at the top of the page `<body>`. It may include a logo, navigational links or a search bar. `<nav>` encapsulates the page's navigational links. It is often placed inside the `<header>` or `<footer>`. `<main>` encapsulates the main content of a page between the header/navigation and the footer areas. `<footer>` includes the page's footer content at the bottom of the `<body>`.

Embedding media

Semantic HTML introduces us to `<video>`, `<audio>` and `<embed>`. `<video>` allows us to add videos to our website. `<audio>` allows us to implement audio into our website. `<embed>` can be used to implement any type of media. These elements are universal in that they all use the `src` attribute to link the source of the content. `<video>` and `<audio>` requires a closing tag while `<embed>` is a self-closing tag.

```
<!--Video Tag-->
<video src="4kvideo.mp4">video not
supported</video>

<!--Audio Tag-->
<audio src="koreanhiphop.mp3"></audio>

<!--Embed tag-->
<embed src="babyyoda.gif"/>
```

`<figure>` and `<figcaption>`

The `<figure>` element is used to encapsulate media such as an image, diagram. or code snippet. The `<figcaption>` element is used to describe the media encapsulated within the `<figure>` element. Developers will normally use `<figcaption>` within the `<figure>` element to group the media and description. This way, if a developer decides to change the position of the media, the description will follow along with it.

```
<figure>
 <img src="qwerty.jpg">
 <figcaption>The image shows the layout
of a qwerty keyboard.</figcaption>
</figure>
```

`<section>` and `<article>`

`<section>` defines elements in a document, such as chapters, headings, or any other area of the document with the same theme. `<article>` holds content that makes sense on its own such as articles, blogs, and comments. Generally developers will use `<section>` to define a theme for the webpage and use `<article>` to write independent content for that theme. This does not mean that `<article>` has to be used with `<section>`.

```
<section>
  <!--defines theme-->
  <h2>Top Sports league in America</h2>
<!--writes independent content relating
to that theme-->
  <article>
    <p>One of the top sports league is
the nba.</p>
  </article>
</section>
```

`<aside>` Aside Element

The `<aside>` element is used to mark additional information that can enhance another element but isn't required in order to understand the main content. Usually, this information would be in a sidebar or a location where it doesn't obstruct the main piece of content. An example of this would be an article that discusses how to take care of a dog and next to the article an ad would appear advertising a dog grooming product.

```
<article>
<!--Main Content-->
</article>
<aside>
<!--Additional information-->
</aside>
```

Print Share