1
HTML Elements

HyperText Markup Language (**HTML**) is a language that annotates text. Annotation of text is done using elements. Using the following `p` element, as an example, we will see how to use HTML:

```
<p>This is an example</p>
```

HTML elements also have attributes that will modify how they are rendered or interpreted. Attributes are added inside of the starting tag. Here is the `class` attribute in a `div` element:

```
<div class="example">This is an example</div>
```

There have been multiple specifications of HTML so far, but we will just look at the most commonly used and important elements of HTML5. HTML5 is the latest official specification, so we will be as future-proof as possible at the time of writing this book. You will want to follow the specifications of HTML as closely as possible. Most browsers are very forgiving and will render your HTML, but when you go beyond the specifications, you can and will run into strange rendering issues.

All HTML elements will have global attributes. The attributes listed for each element in the sections that follow are the attributes beyond the global attributes.

DOCTYPE

The DOCTYPE element defines the document type of the file, as follows:

```
<!DOCTYPE documentType>
```

Attributes

The `documentType` attribute that you can see in the preceding code lets the browser know the type of document you will use.

Description

HTML5 has a simple document type declaration, `<!DOCTYPE html>`. This lets the browser know that the document is HTML5. The previous versions of HTML needed a formal definition of the version being used, but HTML5 has removed this for simplicity.

Most browsers will enforce strict adherence to the document type declared and try and figure out what it is based on looking at the document. This can lead to rendering issues, so it is recommended that you do follow the standards.

Here is an HTML5 declaration:

```
<!DOCTYPE html>
```

html

The `html` element is the root element of the HTML document:

```
<html manifest></html>
```

Attributes

The `manifest` attribute links to a resource manifest that lists which files should be cached.

Description

The `html` element must directly follow the `DOCTYPE` element. This is the root element that all other elements must be descendants of.

The `html` element must have one `head` element and one `body` element as its children. All other elements will be inside these tags.

Here is what a simple HTML page looks like:

```
<!DOCTYPE html>
<html manifest="offline.appcache"><head>
</head>
<body>
    Hey
</body>
</html>
```

Document metadata

The next elements will give metadata about the document. In addition to this, you can also include links to resources, such as CSS and JavaScript.

head

The `head` element is the metadata parent element. All other metadata elements will be children of this element:

```
<head></head>
```

Description

The `head` element usually must have a `title` element inside it. The elements that can go in `head` are `title`, `link`, `meta`, `style`, `script`, `noscript`, and `base`. These elements are explained next.

Here is an example of the `head` element that defines a `title` and `stylesheet` element:

```
<head>
    <title>Title that appears as the tab name</title>
    <link href="style.css" rel="stylesheet"
      type="text/css" media="all" />
</head>
```

title

The `title` element displays the title of the document. This is what is displayed in the browser's tab or the browser window:

```
<title></title>
```

Description

The `title` element is an aptly named element. This is a required element in `head`, and there should only be one `title` element for a document. Here is a simple example of `title` element:

```
<head>
    <title>Example</title>
</head>
```

link

The `link` element links a resource to the current document:

```
<link crossorigin href media rel sizes type></link>
```

Attributes

The attributes that are used in the `link` element are as follows:

- `crossorigin`: This tells the browser whether to make the **Cross-Origin Resource Sharing** (**CORS**) request or not
- `href`: This indicates the URL of the resource
- `media`: This selects the media that this resource applies to
- `rel`: This indicates the relationship of this resource to the document
- `sizes`: This is used with `rel` when it is set to `icons`
- `type`: This indicates the type of content of the resource

Description

The `link` element has a lot of attributes, but most of the time, it is used for loading the CSS resources. This means that you will want to use the attributes `href`, `rel`, `type`, and `media` at least.

You can have multiple `link` elements in a `head` element. Here is a simple example of how to load CSS:

```
<link href="style.css" rel="stylesheet"
  type="text/css" media="all" />
<link href="style.css" media="screen"
  rel="styleshhet" sizes type="text/css"></link>
```

See also

You can also refer to the `crossorigin`, `href`, `media`, `rel`, `sizes`, and `type` attributes to find out more details about the `title` element.

meta

The `meta` element contains metadata about the document. The syntax is as follows:

```
<meta content http-equiv name></meta>
```

Attributes

The attributes that are used in the `meta` element are as follows:

- `content`: This states the value of either the `name` or `http-equiv` attribute.
- `http-equiv`: This attribute, in the case of HTML5, can be set to `default-style`, which sets the default style. Alternatively, it can be set to `refresh`, which can specify the number of seconds taken to refresh the page and a different URL for the new page if needed, for example, `http-equiv="1;url=http://www.google.com"`.
- `name`: This states the name of the metadata.

Description

The `meta` tag has many nonstandard applications. The standardized applications can be viewed in *Chapter 2*, *HTML Attributes*.

Apple has many `meta` tags that will pass information to an iOS device. You can set a reference to an App Store application, set an icon, or display the page in the full screen mode, as just a few examples. All of these tags are nonstandard, but useful when targeting iOS devices. This is true for many other sites and companies.

You can use multiple `meta` tags in a `head` element. Here are two examples. The first example will refresh the page every 5 seconds and the other will define the `author` metadata:

```
<meta http-equiv="refresh" content="5" />
<meta name="author" content="Joshua" />
```

See also

You can also refer to the `name` attribute to find out more details about the `meta` element.

style

The `style` element contains the style information.

CSS:

```
<style media scoped type></style>
```

Attributes

The attributes that are used in the `style` element are as follows:

- `media`: This is a media query
- `scoped`: The styles contained in this element only apply to the parent element
- `type`: This sets the type of document; the default value is `text/css`

Description

The `style` element is usually in the `head` element. This allows you to define CSS styles for the current document.

The preferred method of using CSS is to create a separate resource and use the `link` element. This allows styles to be defined once and used everywhere on a website instead of defining them on every page. This is a best practice as it allows us to define the styles in one spot. We can then easily find and change styles.

Here is a simple inline stylesheet that sets the font color to blue:

```
<style media="screen" scoped type="text/css">
    body{
        color: #0000FF;
    }
</style>
```

See also

You can also refer to the global attributes and Chapters 3-7 to know more details about the style element.

base

The `base` element is the base URL for the document. The syntax is as follows:

```
<base href target>
```

Attributes

The attributes that are used in the `base` element are as follows:

- `href`: This indicates the URL to be used as the base URL
- `target`: This indicates the default target to be used with the URL

Description

The base URL is used whenever a relative path or URL is used on a page. If this is not set, the browser will use the current URL as the base URL.

Here is an example of how to set the base URL:

```
<base href="http://www.packtpub.com/">
```

See also

You can also refer to the `target` attribute to find out more details about the `base` element.

script

The `script` element allows you to reference or create a script for the document:

```
<script async crossorigin defer src type><script>
```

Attributes

The attributes that are used in the `script` element are as follows:

- `async`: This is a Boolean attribute that tells the browser to process this script asynchronously. This only applies to the referenced scripts.
- `crossorigin`: This tells the browser whether to make a CORS request or not.
- `defer`: This is a Boolean attribute that tells the browser to execute the script after the document has been parsed.
- `src`: This distinguishes the URL of the script.
- `type`: This defines the type of the script that defaults to JavaScript if the attribute is omitted.

Description

The `script` element is the way to get JavaScript into your document and add enhanced interactivity. This can be done using a bare `script` tag and adding JavaScript into the element. Also, you can use the `src` attribute to reference an external script. It is considered a best practice to reference a script file as it can be reused here.

This element can be a child of `head` or can be placed anywhere in the body of the document. Depending on where the `script` element is located, you may or may not have access to the DOM.

Here are two examples of using a `script` element. The first example will reference an external script, the second will be an inline `script` element, and the last will show how to use the `crossorigin` attribute:

```
<script src="example.js" type="text/javascript"></script>
<script>
    var a = 123;
</script>
<script async crossorigin="anonymous" defer
  src="application.js" type="text/javascript"><script>
```

noscript

The `noscript` element will be parsed if scripting is turned off in the browser. The syntax is as follows:

```
<noscript></noscript>
```

Description

If scripting is enabled, the content inside of this element will not appear on the page and the code inside it will run. If scripting is disabled, it will be parsed.

This element is mainly used to let the user know that the site may not work with JavaScript. Almost every website today not only uses JavaScript, but requires it.

Here is an example of the `noscript` element:

```
<noscript>
    Please enable JavaScript.
</noscript>
```

Semantic content sections

The next elements are the main elements to use when adding content to the document. For example, using the `article` element instead of an arbitrary `div` element allows the browser to infer that this is the main content of the page. These elements should be used to give structure to a document and not be used for styling purposes. Semantic elements make our HTML document more accessible using an ever-increasing amount of different devices.

body

The `body` element is the main content section of the document. There must be only one main element, its syntax is as follows:

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

Attributes

The attributes of the `body` element include the `inline` event attributes.

Description

The `body` element is the main content section of most documents. It must be the second child element of `html`, and there should only be one `body` element in a document.

Here is an example of the `body` element:

```
<body>
    <span>Example Body</span>
</body>
```

section

The `section` element describes the content section of a document. For example, this can be a chapter of a book:

```
<section></section>
```

Description

The `section` element is a new element that was introduced in HTML5. A `section` element should group the content together. While not a requirement, using a `heading` element as the first element in the code is a best practice. The section should be viewed as another part of the outline of the document. It groups related items into an easily targeted area. You can use this element multiple times in a document.

Here is an example of the `section` element:

```
<section>
    <h2>Section Heading</h2>
    <p>Section content.</p>
</section>
```

nav

The `nav` element is the navigation element:

```
<nav></nav>
```

Description

The `nav` element is another semantic element introduced with HTML5. This lets the browser know that the content of this element is the parent and is for navigation. The `nav` element enhances accessibility by giving screen readers a landmark for navigation. This element should wrap any site navigation or other links that are grouped together for ease of navigation. You can use this multiple times.

Here is an example of using the `nav` element:

```
<nav>
    <ul>
        <li><a href="#">Home</a></li>
    </ul>
</nav>
```

article

The `article` element is designed to wrap content that can stand on its own:

```
<article></article>
```

Description

The `article` element is a new element introduced in HTML5. The `article` element is similar to the `section` element; in that, it denotes that the content in the element is the core part of the page. The `article` element should be a complete composition that can stand on its own. For example, in a blog, the actual blog post should be wrapped with an `article` element.

Content can then be further broken down using either an `article` element or a `section` element. There is no standard rule for when to use either. However, both should be related to the content in the outer `article` element.

Here is an example of the `article` element being used:

```
<article>
    <header>
        <h1>Blog Post</h1>
    </header>
    <p>This post covers how to use an article element...</p>
    <footer>
        <address>
            Contact the author, Joshua
        </address>
    <footer>
</article>
```

Headings

The `heading` elements are the elements that specify different levels of headings according to their importance:

```
<h1></h1>
<h2></h2>
<h3></h3>
<h4></h4>
<h5></h5>
<h6></h6>
```

Description

These should be used to give relative importance to different headings. For example, `h1` should be used for the title of the document. The importance of a heading goes down as the heading value goes up, that is, `h6` is the least important level of heading in the example that follows.

Here is an example using all the headings:

```
<h1>Heading Importance 1</h1>
<h2>Heading Importance 2</h2>
<h3>Heading Importance 3</h3>
<h4>Heading Importance 4</h4>
<h5>Heading Importance 5</h5>
<h6>Heading Importance 6</h6>
```

See also

You can also refer to the global attributes to learn the `heading` element in more detail.

header

The `header` element groups the content that is considered to be the header for a particular group of content, its syntax is as follows:

```
<header></header>
```

Description

The `header` element is usually one of the first content elements on the page. It will most likely contain navigation options, a logo, and/or a search box. The header is usually repeated on multiple pages of a website. Each section or article can also contain a header. This is a new element introduced in HTML5.

Here is an example of the `header` element:

```
<header>
    <img src="logo.png" />
</header>
```

See also

You can also refer to the global attributes to find out about the `header` element in more detail.

footer

The `footer` element provides a footer of a particular group of content, its syntax is as follows:

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

Description

The `footer` element wraps all the content that is considered to be the footer of the document. Usually, this will include copyright, author, and/or social media links. Of course, what you decide to put into a footer is arbitrary. Each section or article can also contain a footer.

Here is an example of the `footer` element:

```
<footer>
    Written by: Joshua Johanan
</footer>
```

address

The `address` element is used for the contact address of the author or organization, its syntax is as follows:

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

Description

Use the `address` element when you have the contact information of the user. The `address` element will add semantic value to the content, contrary to a regular `div` element.

Usually, this will be placed in the footer, but it can be used in an article or body section. It will apply to the nearest `article` element or to the entire document. Do not use any of the content section elements in an `address` element.

Here is the `address` element in use:

```
<footer>
    <address>
        Please contact me at my <a href="#">website</a>
    </address>
</footer>
```

aside

The `aside` element is for supplemental content:

```
<aside></aside>
```

Description

Use the `aside` element to highlight the content that is tied to the main article. Some examples in the context of the blog would be the author's profile, other posts by this author, or even related advertisements.

Here is an example of `aside`:

```
<aside>
    Peyton Manning is a 5-time MVP (2003, 2004, 2008, 2009, 2013)

</aside>
```

p

The `p` element is known as the paragraph element. This is a block element, its syntax is as follows:

```
<p></p>
```

Description

The `p` element should be used to distinguish between separate paragraphs in a document. This element is associated with the `text` and `inline` elements. You will not want to use a `div` element, for example. If you find yourself wanting to do this, you may want to build your document differently.

Here are a couple of paragraphs:

```
<p>This is an intro paragraph.</p>
<p>This paragraph will build upon the opening.</p>
```

Content sections

The `content` sections are quite similar to the semantic content sections. The main difference is that the use of all the given elements are not driven by the outline or purpose of the document like the semantic sections are.

hr

The `hr` element is the horizontal rule element, its syntax is as follows:

```
<hr>
```

Description

By default, the `hr` element will draw a horizontal line in the content. You can change the look of this element through CSS.

This element should never have any content inside of it:

```
<p>This is a paragraph.</p>
<hr/>
<p>This paragraph goes in another direction.</p>
```

pre

The `pre` element is the preformatted text:

```
<pre></pre>
```

Description

The text in an HTML document is usually not shown in the browser with the same whitespace or line breaks as it is in a text document. The `pre` element allows you to display text in the same way as it is in the document. Whitespace and line breaks will be preserved.

Here is an example of using line breaks:

```
<pre>This text
has some
line breaks.</pre>
```

blockquote

The syntax of a `blockquote` element is as follows:

```
<blockquote cite></blockquote>
```

Attributes

The `cite` attribute is used in the `blockquote` element to point to the URL of the cited document.

Description

The `blockquote` element is used when pulling a quotation out of a document or text.

Here is an example:

```
<blockquote cite="https://www.packtpub.com/">
    <p>Contact Us</p>
</blockquote>
```

ol

The `ol` element is the ordered list element, which has the following syntax:

```
<ol reversed start type></ol>
```

Attributes

The attributes that are used in the `ol` element are as follows:

- `reversed`: This is a Boolean value. It denotes that the list is in a reverse order.
- `start`: This accepts a value as a number to start with.
- `type`: This will change the type of the numbered elements. By default, we can have a numbered list (`1`), but we can change to other types, such as lowercase letters (`a`), uppercase letters (`A`), lowercase Roman numerals (`i`), and uppercase Roman numerals (`I`).

Description

The `ol` element can be used in the same situations as a `ul` element, except that an `ol` element is numbered instead of bulleted. For example, you would use a `ul` element for a grocery list and an `ol` element for a numbered set of instructions. You can have multiple `ul` or `ol` elements nested within each other.

The items in the list will be the `li` elements.

Here is an example of a list that uses Roman numerals and starts at `10`.

```
<ol start="10" type="i">
    <li>Roman numeral 10</li>
    <li>Roman numeral 11</li>
</ol>
```

See also

You can also refer to the `ul` and `li` elements to find out more about the `ol` element.

ul

The `ul` element is an unordered list element:

```
<ul></ul>
```

Description

The `ul` element can be used in the same situations as an `ol` element, but a `ul` element will be bulleted and an `ol` element will be numbered.

When you style this list, you should use CSS and not the older HTML 4 attributes.

You can nest the `ul` and `ol` elements multiple times.

Here is an example of the `ul` element:

```
<ul>
    <li>Items in</li>
    <li>no particular</li>
    <li>order</li>
</ul>
```

See also

You can also refer to the `ol` and `li` elements to learn more about the `ul` element.

li

The `li` element is the list item element:

```
<li value></li>
```

Attributes

The `value` attribute is used in the `li` element with the `ol` element and it is the value of the item in the ordered list.

Description

You will use the `li` element for each item in a list.

Here is an example:

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

See also

You can also refer to the `ol` and `ul` elements to know more details about the `li` element.

dl

The `dl` element is the definition list element:

```
<dl></dl>
```

Description

The `dl` element is a list where the items have a term and definition; however, the `dl` element can be used for more than just terms and definitions.

You must use the `dt` element followed by the `dd` element when building the list for the `dl` element. Each `dt` element can have multiple `dd` elements after it.

Here is an example:

```
<dl>
    <dt>PactPub</dt>
    <dd>Packt Publishing</dd>
</dl>
```

See also

You can also refer to the `dt` and `dd` elements to find out more about the `dl` element.

dt

The `dt` element is the definition term element:

```
<dt></dt>
```

Description

The `dt` element is the first item in a definition list, the `dd` element being the other item.

Here is an example:

```
<dl>
    <dt>PactPub</dt>
    <dd>Packt Publishing</dd>
</dl>
```

See also

You can also refer to the `dl` and `dd` elements to find out more about the `dt` element.

dd

The `dd` element is the definition description element:

```
<dd></dd>
```

Description

The `dd` element is the second item in a definition list, the other one being the `dt` element.

Here is an example:

```
<dl>
    <dt>PactPub</dt>
    <dd>Packt Publishing</dd>
</dl>
```

See also

You can also refer to the `dl` and `dd` elements to find out more about the `dd` element.

figure

The syntax of the `figure` element is as follows:

```
<figure></figure>
```

Description

The `figure` element is a new element introduced with HTML5. In much the same way as an article adds some meaning where there was none, a figure adds meaning too. A figure is an image or any other item of information that is related to the document. This has more meaning than just using an `img` element.

Here is an example:

```
<figure>
     <img src="figure1.jpg" title="Figure 1" />
     <figcaption>Figure One</figcaption>
</figure>
```

See also

You can also refer to the `figcaption` element to find out more about the `figure` element.

figcaption

The `figcaption` element is the figure caption element:

```
<figcaption></figcaption>
```

Description

The `figcaption` element was introduced in HTML5 along with the figure. This element provides the caption for a figure. This element must be inside a `figure` element and it must be either the first or last child of the `figure` element.

Here is a simple example of the `figcaption` element:

```
<figure>
     <img src="figure1.jpg" title="Figure 1" />
     <figcaption>Figure One</figcaption>
</figure>
```

See also

You can also refer to the `figure` element to find out more about the `figcaption` element.

div

The `div` element is the division element:

```
<div></div>
```

Description

The `div` element is one of the most used elements in HTML today. It is the element used to split up your document into arbitrary divisions. The `div` element has no default styling. These divisions could be for placement, styling, or any other reason. A `div` element does not affect the semantic meaning of the document. It should only be used when no other element suits your requirements.

Here is an example:

```
<div>
    You can put whatever you want in here!
    <div>
        More elements.
    </div>
</div>
```

main

The syntax of the `main` element is as follows:

```
<main></main>
```

Description

The `main` element should have the main content of the document inside it. You cannot have this element as a descendant of the `article`, `aside`, `footer`, `header`, or `nav` elements. This differs from an article, in that, an article should be a self-contained element.

Here is an example of the `main` element in use:

```
<main>
    This is the main content of the document.
    <article>
        Here is the article of the document.
    </article>
</main>
```

Inline elements

The following elements can all wrap text- and block-level elements to give them functionality, style, and meaning.

a

The `a` element is the anchor element. This is where HTML gets the **HyperText** (**HT**), the syntax is as follows:

```
<a download href media ping rel target type></a>
```

Attributes

Here are the attributes that are used in the `a` element:

- `download`: This attribute lets the browser know that the item should be downloaded. The dialog will default to the filename in this attribute.
- `href`: This is the link target.
- `media`: This states the media that the stylesheet should apply to based on a media query.
- `ping`: This makes a URL to ping and notify if the link is followed.
- `rel`: This states the relationship of the document being linked to.
- `target`: This states where the target link is to be displayed.
- `type`: This states the MIME type of the linked resource.

Description

The `a` element is one of the most important and useful elements. It allows you to link documents together and you can easily jump between elements. We can say that the Web would not be as popular as it is now without this very easy-to-use element.

The link can be that of an `anchor` tag in the document, a relative URL, or any external resource. When linking to an `anchor` tag in the current document, use the `a` tag and the `id` attribute.

The content you put inside the `a` element will become part of what the user can click on to follow the link. This includes the `text`, `img`, and `div` elements, to name a few.

Here is an example of an `img` element with an image:

```
<a href="http://www.packtpub.com">
    <img src="packt_logo.png" />
</a>
```

Here is an example of a PDF document that will be downloaded; this will track each click:

```
<a download="report.pdf" href="assests/report.pdf"
  media="min-width: 1024px" ping="track/click" rel="alternate"
  target="_blank" type=" application/pdf"></a>
```

abbr

The `abbr` element is the abbreviation element:

```
<abbr></abbr>
```

Description

The `abbr` element is used to show what an abbreviation stands for. You should put the full word in the `title` attribute. Most browsers will display this as a tooltip when you hover over this element.

Here is an example:

```
<abbr title="abbreviation">abbr</abbr>
```

bdo

The `bdo` element is the bi-direction override element:

```
<bdo dir></bdo>
```

Attributes

The `dir` attribute is used in the `bdo` element, which gives the direction of the text. Its values can be `ltr`, `rtl`, and `auto`.

Description

The `bdo` element will override the current direction of the text for the direction defined in the element.

Here is an example:

```
<bdo dir="rtl">Right to Left.</bdo>
```

br

The `br` element is the line break element:

```
<br>
```

Description

The `br` element adds a line break. This is needed as line breaks in text are ignored when rendered in the browser. This should not be used to help place elements, as that is the job of CSS.

Here is an example:

```
First Name<br>
LastName
```

cite

The `cite` element is the citation element:

```
<cite></cite>
```

Description

The `cite` element is used to cite another source. Most browsers will render this in italics.

Here is an example:

```
This quote is from <cite>Web Developer's Reference</cite>
```

code

The syntax of the `code` element is as follows:

```
<code></code>
```

Description

The `code` element is used to display the programming code in a document. The browser will use a monospace font for it.

Here is an example:

```
Here is some JavaScript: <code>var a = 'test'</code>
```

dfn

The `dfn` element is the defining instance element:

```
<dfn></dfn>
```

Description

The `dfn` element is used to create a defining instance or the first time a specific word is introduced and explained.

Here is an example of the `dfn` element:

```
<dfn>HTML</dfn>, or HyperText Markup Language.
```

em

The `em` element is the emphasis element:

```
<em></em>
```

Description

The `em` element is used to add more emphasis to a specific word or phrase. By default, browsers will render this in italic font, but it should not just be used for italics.

kbd

The `kbd` element is the keyboard input element:

```
<kbd></kbd>
```

Description

The `kbd` element is used for text that the user should input. This does not mean that the user inputs data into the element, rather they will enter it into a window, console, or some application on their computer.

Here is an example:

```
Press <kbd>Win + R</kbd> to open the Run dialog.
```

mark

The syntax of the `mark` element is as follows:

```
<mark></mark>
```

Description

The `mark` element is used to highlight text.

Here is an example:

```
<mark>This</mark> will be highlighted
```

q

The `q` element is the quote element:

```
<q cite></q>
```

Attributes

The `cite` attribute used in the `q` element states the URL of the source of the quote.

Description

The `q` element is used for short quotes. For longer quotes, use `blockquote`.

Here is an example:

```
<q cite="http://en.wikiquote.org/">Don't quote me on this.</q>
```

See also

You can also refer to the `blockquote` attribute to learn more about the `q` element.

s

The `s` element is the strikethrough element:

```
<s></s>
```

Description

The `s` element should be used when a piece of information in the document is no longer accurate. This is different than a revision made to the document.

Here is an example:

```
Today is the <s>twenty-fifth<s> twenty-sixth.
```

samp

The `samp` element is the sample output element:

```
<samp></samp>
```

Description

The `samp` element is used to show the sample output from a command or program.

Here is an example:

```
The command should output <samp>Done!</samp>
```

small

The syntax of the `small` element is as follows:

```
<small></small>
```

Description

The `small` element is used to make the text smaller. This is usually done with text such as the copyright or legal text.

Here is an example:

```
<small>Copyright 2014</small>
```

span

The syntax of the `span` element is as follows:

```
<span></span>
```

Description

The `span` element is much like the `div` element; in that, it is just an arbitrary container. The `div` element is a block-level element, and the `span` element is an inline element. The element does not add any semantic meaning to the text or document. Often, it is used to add a CSS style to the text:

```
<span>This text is in the span element.</span>
```

strong

The syntax of the `strong` element is as follows:

```
<strong></strong>
```

Description

The `strong` element should be used when certain text needs more importance. This carries some semantic meaning. The `strong` element's default style in most browsers is bold. This should not then be interchangeable with the `b` element, as that does not carry any semantic meaning.

Here is an example:

```
<strong>Warning!</strong> JavaScript must be enabled.
```

sub

The `sub` element is the subscript element:

```
<sub></sub>
```

Description

The `sub` element will render the text as a subscript.

Here is an example:

```
H<sub>2</sub>O
```

sup

The `sup` element is the superscript element:

```
<sup></sup>
```

Description

The `sup` element will render the text as a superscript.

Here is an example:

```
x<sup>2</sup> is what x squared should look like
```

time

The syntax of the `time` element is as follows:

```
<time datetime></time>
```

Attributes

The `datetime` attribute used in the `time` element gives a string that is the date and time value.

Description

The `datetime` element allows browsers to easily parse dates out of a document. You can wrap a date or the description of a date (tomorrow or July 4, for example) and still have an exact date for the browser to read.

Here is an example:

```
The party is on <time datetime="2014-11-27 14:00">
  Thanksgiving @ 2PM</time>
```

var

The `var` element is the variable element:

```
<var></var>
```

Description

The `var` element is used for variables in a mathematical expression or for programming.

Here is an example:

```
The variable <var>x</var> is equal to the string test
  in this example.
```

wbr

The `wbr` element is the word break opportunity element:

```
<wbr>
```

Description

The `wbr` element is a new element that was introduced with HTML5. We use this element to let the browser know of a good spot to break between words. This does not force a break, but if a break is needed, then the browser will respect the element.

It is an empty tag, meaning that it should not have an ending tag.

Here is an example:

```
If you have a really short width <wbr>then you <wbr>
  could have breaks.
```

Embedded content

The following elements are used to embed media or other objects into the document.

img

The `img` element is the image element:

```
<img alt crossorigin height ismap sizes src srcset width />
```

Attributes

The attributes that are used in the `img` element are as follows:

- `alt`: This is the alternate text for the image. Use this to describe the image. This is used to enhance accessibility.
- `crossorigin`: This lets the browser know whether this image should be fetched with a CORS request. If the image will be modified in a canvas element and not from the same domain name, then a CORS request must be used.
- `height`: This is an attribute to set the height of the image.
- `ismap`: This lets the browser know whether the image is used in a server-side map.
- `sizes`: This is a list of media conditions that will map to a size. This is used to help the browser determine which image to use. By default, this will be 100 VW, which is 100% of the view width.
- `src`: This is the most important attribute, it is the image URL.
- `srcset`: This is a list of multiple images that can be used for display on our web page. This is used to target different screen sizes or pixel densities.
- `width`: This is an attribute to set the width of the image.

Description

The `img` element is used if you want an image in the document. This element has many attributes, but the `src` and `alt` attributes are the only required attributes. The `alt` attribute should be used to describe the image in almost 100% of the cases. The main exception is when the image is only used as a decorative image, for example, when an image is used instead of a horizontal rule. The width and height can be used if the image is of a different size than what is needed on the page; otherwise, it defaults to the size of the image.

The `crossorigin` element can be confusing. It is used to ensure that you have ownership of an image before you modify the image in a canvas element. The image needs to either be from the same fully qualified domain name or the server's response must let the browser know whether the current domain can use the image.

Finally, `srcset` is used to give the browser a list of images that it can use. This is done with a comma-separated list of URLs and a descriptor. A descriptor can either be a width descriptor, which would be a number followed by `w`, or a pixel descriptor, which is a number followed by `x`. The width descriptor tells the browser the width of the image. The pixel descriptor tells the browser the pixel density it should use for the image.

The width descriptor can also be used by the browser when the pixel density changes. For example, if you have an image that is double the resolution and the pixel density doubles as well, the browser will choose the larger resolution.

The `sizes` element is used along with `srcset` to help the browser identify a break point. This is done using a media condition, for example, `"(min-width: 1600px) 25vw, 100vw"`. This states that if the width of the page is at least `1600` pixels, the images will be 25% of the view width, otherwise the view width is 100%. This helps the browser know where you want a break point and how large you want the image to be.

The best way to think about `srcset` is that you are letting the browser know about all the images that can be used in a specific `img` element. You include the information that the browser is most concerned about—width and pixel density—and let the browser choose.

Here are a few examples. The first example is a simple `img` tag and the next one uses `crossorigin`:

```
<img src="example.png" alt="This is an example image"/>
<img src="example.png" crossorigin="anonymous"
  alt="This is an example image"/>
```

Here is a `srcset` example that lets the browser choose the image based on pixel density:

```
<img src="normal.jpg" srcset="normal.jpg 1x,
  retina.jpg 2x" alt="Image of Person in article" />
```

The following is an example using `srcset` and widths:

```
<img src="regular.jpg" srcset="double_size.jpg 1000w,
  regular.jpg 500w" sizes="100vw" alt="A bird"/>
```

iframe

The `iframe` element is the inline frame element:

```
<iframe height name seamless src width></iframe>
```

Attributes

The attributes that are used in the `iframe` element are as follows:

- `height`: This is the attribute to set the height.
- `name`: This states a name that can be used in the target attribute.
- `seamless`: This makes `iframe` appear as part of the content of the document. This will apply the outer context CSS and enables us to open links in the outer context.
- `src`: This is the URL of the embedded document.
- `width`: This is the attribute to set the width.

Description

The `iframe` element is used to embed another full HTML document inside the current document.

Here is an example that loads the Google homepage and another that loads Packt Publishing's page:

```
<iframe src="http://www.google.com"></iframe>
<iframe height="100px" name="remote-document"
  seamless src="https://www.packtpub.com/" width="100px"></iframe>
```

embed

The syntax of the `embed` element is as follows:

```
<embed height src type width/>
```

Attributes

The attributes that are used in the `embed` element are as follows:

- `height`: This is the attribute to set the height
- `src`: This is the URL of the object to be embedded
- `type`: This is the MIME type of the object
- `width`: This is an attribute to set the width

Description

The `embed` element is used to embed other objects in the document. There are other elements for embedding objects as well, depending on their type. For example, you can embed a video using the video element, as follows:

```
<embed src="example.mp4" type="video/mp4"/>
```

See also

You can also refer to the `audio`, `video`, and `object` elements to learn more about the `embed` element.

object

The syntax of the `object` element is as follows:

```
<object data height type width></object>
```

Attributes

Here are the attributes that are used in the `object` element:

- `data`: This is the URL of the object to be embedded
- `height`: This is the attribute to set the height
- `type`: This is the MIME type of the object
- `width`: This is the attribute to set the width

Description

The `object` element can be used very much like the `embed` element. This has historically been used for the `Flash` objects.

Here is an example:

```
<object data="example.swf" type="application/x-shockwave-flash">
  </object>
```

See also

You can also refer to the `audio`, `video`, `embed`, and `param` attributes to find out more about the `object` element.

param

The `param` element is the parameter element:

```
<param name="movie" value="video.swf"/>
```

Attributes

The attributes that are used in the `param` element are as follows:

- `name`: This is the name of the parameter
- `value`: This is the value of the parameter

Description

The `param` element defines a parameter for the object element. The parent of this element should be an `object` element.

Here is an example. This example is useful when using objects on older browsers:

```
<object data="example.swf" type="application/x-shockwave-flash">
    <param name="movie" value="example.swf" />
</object>
```

video

The syntax of the `video` element is as follows:

```
<video autoplay buffered controls crossorigin
  height loop muted played poster src width></video>
```

Attributes

The attributes that are used in the `video` element are as follows:

- `autoplay`: This is a Boolean attribute that tells the browser to start playing the video as soon as it can play after the loading has stopped
- `buffered`: This is a `read` object that tells how much of the video has buffered
- `controls`: This is a Boolean attribute to decide whether to display the controls
- `crossorigin`: This attribute is used make a CORS request if you plan on modifying the video in a canvas and the video is not hosted at the same fully qualified domain name

- `height`: This is the attribute to set the height
- `loop`: This states whether or not to loop the video
- `muted`: This states whether or not to mute the audio
- `played`: This is a `read` object, which reads how much of the video has been played
- `poster`: This is a URL of an image that will be displayed until the video can played
- `src`: This is the URL of the video
- `width`: This is the attribute to set the width

Description

The `video` element is a new element introduced in HTML5. You can use this to play a video directly in the browser. This is very useful as the user does not need a plugin or special player to view the video. In addition to this, you can use the `video` element as a source for the `canvas` element.

You can also use the `source` element to include multiple sources in case the browser can only play a certain type of file. If the browser does not support the `video` element or the file type, you can put the fallback content into the element.

Here is an example using the `video` element and another that demonstrates possible values for all of the attributes:

```
<video src="example.mp4" autoplay poster=
  "example-poster.png"></video>
<video autoplay buffered controls crossorigin="anonymous"
  height="100px" loop muted played poster="cover.jpg"
  src="video.ogg" width="100px"></video>
```

See also

You can also refer to the `source` and `track` attributes to find out more about the `video` element.

audio

The syntax of the `audio` element is as follows:

```
<audio autoplay buffered controls loop muted played
  src volume></audio>
```

Attributes

The attributes that are used in the `audio` element are as follows:

- `autoplay`: This is the attribute in which the browser will start playing the audio as soon as it can without loading
- `buffered`: This is the attribute that has the buffered time ranges
- `controls`: This is the attribute that has the browser display controls
- `loop`: This is the attribute that decides whether or not to loop the audio
- `muted`: This is the attribute that decides whether or not to mute the audio
- `played`: This is the attribute that has the time ranges of the audio played
- `src`: This is the attribute that gives the URL of the audio
- `volume`: This is the attribute that ranges the volume from 0.0 to 1.0

Description

The `audio` element was introduced in HTML5. You can add an audio to your page and make the browser play it.

You can also use the `source` element to include multiple sources in case the browser can play a certain type of file. If the browser does not support the audio element or the file type, you can put fallback content into the element.

Here is an example using the `audio` element:

```
<audio src="test.mp3" autoplay loop>
    Your browsers does not support the audio element.
</audio>
```

See also

You can also refer to the `source` and `track` attributes to find out more about the `audio` element.

source

The syntax of the `source` element is as follows:

```
<source src type />
```

Attributes

The attributes that are used in the `source` element are as follows:

- `src`: This is the URL of the resource
- `type`: This is the MIME type of the resource

Description

The `source` element is used to add multiple sources to the `audio`, `picture`, and `video` elements. It must be a child of one of these elements. You can use this to specify multiple formats of the same resource. For example, you can have two different video encodings for a video. If the browser cannot play the first, it will fall back to the other.

Here is an example with an `audio` element:

```
<audio autoplay controls>
    <source src="test.ogg" type="audio/ogg" />
    <source src="test.mp3" type="audio/mpeg">
</audio>
```

See also

You can also refer to the `audio` and `video` attributes to find out more about the `audio` element.

track

The syntax of the `track` element is as follows:

```
<track default kind label src />
```

Attributes

Here are the attributes that are used in the `track` element:

- `default`: This states whether the chosen track is the default track or not.
- `kind`: This states the different kinds of tracks that can be loaded. Here are the values: `subtitles`, `captions`, `descriptions`, `chapters`, or `metadata`.
- `label`: This is the title of the track.
- `src`: This is the URL of the resource.

Description

You will mainly use the `track` element to add captions or subtitles to videos.

Here is an example video with captions:

```
<video src="test.mp4">
    <track label="English" kind="captions" src="en.vtt" default>
    <track label="Spanish" kind="captions" src="sp.vtt">
</video>
```

Tables

Tables are useful for showing data. They make defining rows and columns very easy. In the past, tables were used to create layouts, but today, that is done with CSS. They should be used to only display the tabular data.

table

The syntax of the `table` element is as follows:

```
<table></table>
```

Description

The `table` element is the root element for creating a table. All the other elements in this section must be children of this element.

Here is a simple example of the `table` element:

```
<table>
    <tr>
        <td>Column in Row 1</td>
    </tr>
</table>
```

caption

The syntax of the `caption` element is as follows:

```
<caption></caption>
```

Description

The `caption` element will be the title of the table. This element must be the first child of the `table` element.

Here is a simple example:

```
<table>
    <caption>Caption for the table</caption>
    <tr>
        <td>Column in Row 1</td>
    </tr>
</table>
```

colgroup

The `colgroup` element is the column group element:

```
<colgroup span></colgroup>
```

Attributes

The `span` attribute states the number of columns the group spans.

Description

The `colgroup` element is used to define styles that are common to all columns or groups of columns. This element is not as useful as it once was as the new CSS selectors can target all the columns and even some specific columns.

tbody

The `tbody` attribute is the table body element:

```
<tbody></tbody>
```

Description

The `tbody` attribute is the main part of a table. All of the data rows and columns should go in this element. This element should have one or more `tr` elements as its children.

Here is an example:

```
<table>
    <tbody>
        <tr>
            <td>Column in Row 1</td>
        </tr>
    </tbody>
</table>
```

thead

The `thead` element is the table head element:

```
<thead></thead>
```

Description

The `thead` element is the row that has all of the column headings. It must appear before the `tbody` or `tfoot` elements.

Here is an example:

```
<table>
    <thead>
        <tr>
            <th>Heading 1</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Column in Row 1</td>
        </tr>
    </tbody>
</table>
```

tfoot

The `tfoot` element is the table footer element:

```
<tfoot></tfoot>
```

Description

The `tfoot` element is the footer for the table. It must be used after any `thead` elements, but can be either before or after `tbody`. The placement of the `tfoot` element does not affect where it is rendered, which is always at the bottom.

Here is an example:

```
<table>
    <tbody>
        <tr>
            <td>Column in Row 1</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td>Footer 1</td>
        </tr>
    </tfoot>
</table>
```

tr

The `tr` element is the table row element:

```
<tr></tr>
```

Description

The `tr` element is the row element. Every time you need another row in a table, use this element. This element can be the child of a `table`, `tbody`, `thead`, or `tfoot` element. You must use either a `td` or `th` as its child.

Here is an example:

```
<table>
    <tbody>
        <tr>
            <td>Column in Row 1</td>
        </tr>
    </tbody>
</table>
```

td

The `td` element is the table cell element:

```
<td colspan headers rowspan></td>
```

Attributes

The attributes that are used in the `td` element are as follows:

- `colspan`: This tells how many columns it will span as an integer
- `rowspan`: This tells how many rows the `rowspan` attribute will span as an integer
- `headers`: This is a space-separated list of strings that match the ID of any `th` element

Description

The `td` element is the basic table column element. The `colspan` and `rowspan` attributes allow you to make the column wider and taller, respectively.

Here is an example:

```
<table>
    <tbody>
        <tr>
            <td>Column in Row 1</td>
        </tr>
    </tbody>
</table>
```

th

The `th` element is the table header cell element:

```
<th colspan rowspan></th>
```

Attributes

The attributes that are used in the `th` element are as follows:

- `colspan`: This states the number of columns the `colspan` attribute will span as an integer
- `rowspan`: This states the number of rows the `rowspan` attribute will span as an integer

Description

The `th` element is used when we add a column to the `thead` element.

Here is an example:

```
<table>
    <thead>
        <tr>
            <th>Header</th>
        </tr>
    </thead>
</table>
```

Forms

Forms are great for getting information from the user. They usually have multiple elements inside them that accept input from the user. The data acquired from the input of the user is then sent to the server to process.

Form

The syntax of the `form` element is as follows:

```
<form accept-charset action autocomplete enctype
  method name novalidate target></form>
```

Attributes

The attributes that are used in the `form` element are as follows:

- `accept-charset`: This is a list of character encodings that the server accepts. This can be a space- or comma-delimited list.
- `action`: This is the URL that will process the form.
- `autocomplete`: This lets the browser know whether it can autocomplete this form with previous values.
- `enctype`: This sets the MIME type for the content being sent to the server.
- `method`: This is the HTTP method that will be used to submit the form. It can be `Post` or `Get`.
- `name`: This is the name of the form.
- `novalidate`: This tells the browser not to validate the form.
- `target`: This states where the response will be displayed. This can be: `_self`, `_blank`, `_parent`, or `_top`.

Description

The `form` element is the root element of a form in a document. When submitted, the form will contain all the data that is entered into the different elements inside of the form.

Here is a simple form of the syntax:

```
<form action="processForm" method="post">
    <input type="text" name="text-input"/>
    <button type="submit">Submit!</button>
</form>
```

fieldset

The syntax of the `fieldset` element is as follows:

```
<fieldset disabled form name></fieldset>
```

Attributes

The attributes that are used in the `fieldset` element are as follows:

- `disabled`: This disables all the elements in the fieldset
- `form`: This is the ID of the form the `form` attribute belongs to
- `name`: This is the name of fieldset

Description

The `fieldset` element allows us to group related inputs together. The default style of most browsers is to put a border around the fieldset.

If the first element is a `legend` element, then the fieldset will use that as its label.

Here is an example of using the `fieldset` element:

```
<form action="processForm" method="post">
    <fieldset>
        <legend>This is a fieldset</legend>
        <input type="text" name="text-input" />
    </fieldset>
</form>
```

See also

You can also refer to the `legend` attribute to find out more about the `fieldset` element.

legend

The syntax of the `legend` element is as follows:

```
<legend></legend>
```

Description

The `legend` element will become the label of the `fieldset` element that it is a child of.

See also

You can also refer to the `fieldset` element to find out more about the `legend` element.

label

The syntax of the `label` element is as follows:

```
<label accesskey for form></label>
```

Attributes

The attributes that are used in the `label` element are as follows:

- `accesskey`: This is a shortcut for the `accesskey` element
- `for`: This is the ID of a form element that this is the label for
- `form`: This is the ID of the form the `form` attribute is associated with

Description

The `label` element is for labeling inputs. You can either put the element in the label or use the `for` attribute. When the label is correctly linked to an input, you can click the label and the cursor will be in the input.

Here is an example that covers each different way of labeling an element:

```
<form action="processForm" method="post">
    <label>First name: <input type="text"
      name="firstName" /></label>
    <label for="lastNameInput">Last name:
      </label><input id="lastNameInput" type="text"
      name="lastName" />
</form>
```

input

The syntax of the `input` element is as shown:

```
<input accept autocomplete autofocus checked disabled form
  formaction formenctype formmethod formnovalidate
  formtarget height inputmode max maxlength min minlength
  multiple name placeholder readonly required size spellcheck
  src step tabindex type value width></input>
```

Attributes

The attributes that are used in the `input` element are as follows:

- `accept`: This is used to specify which file types are accepted for the web page.
- `autocomplete`: This says whether the browser can autocomplete this input based on previous values.
- `autofocus`: This lets the browser automatically focus on the element. This should only be used on one element.
- `checked`: This is used with the radio or checkbox. This will select the value on page load.
- `disabled`: This says whether or not to disable the element.
- `form`: This states the ID of the form.
- `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget`: These will override the form's value if these attributes are associated with a button or image.
- `height`: This is used to set the height of the image.
- `inputmode`: This gives a hint to the browser of what keyboard to display. For example, you can use numeric to specify only the keypad.
- `max`: This is the maximum number or date-time of the system.

- `maxlength`: This is the maximum number of characters that can be accepted in the web page.
- `min`: This is the minimum number or date-time of the system.
- `minlength`: This is the minimum number of characters.
- `multiple`: This says whether there can be multiple values or not. This is used with `email` or `file`.
- `placeholder`: This is the text that is displayed in the element when there is no value assigned to this attribute.
- `readonly`: This makes the element of the read-only format.
- `required`: This is the element that is required to be assigned a value and cannot be blank.
- `size`: This is the size of the element.
- `src`: This will be the URL of the image if it is of the `img` type.
- `step`: This is used with the `min` and `max` attributes to determine the incremental steps.
- `tabindex`: This is the order of the elements when using tab.
- `type`: Refer to the next section for the description.
- `value`: This is the initial value of the element.
- `width`: This is the attribute to set the width.

Description

The `input` element is the main way to get data from the user. This element can vary quite a bit based on the type that is used. HTML5 has added a few inputs that also give you validation. For example, the `email` type will also validate that the e-mail is a valid `email`. In addition to this, the type can give hints to the browser about what keyboard to display. This is important for mobiles, which have many different virtual keyboards. For example, the `tel` type will show a number pad instead of the regular keyboard. Here is a rundown of the different types of keyboards and their description:

- `button`: This is a button.
- `checkbox`: This is a checkbox.
- `color`: For most browsers, this will create a color picker; however, it is not required by HTML5.
- `date`: This creates a date picker.

- `datetime`: This creates a date and time picker using a time zone.
- `datetime-local`: This creates a date and time picker without a time zone.
- `email`: This is a text input for e-mail addresses. This type validates e-mails.
- `file`: This selects a file.
- `hidden`: This attribute will not be displayed, but the value will still be part of the form.
- `image`: This essentially creates an image button.
- `month`: This can enter the month and year.
- `number`: This is used for a floating point number.
- `password`: This is a text input where the text is not shown.
- `radio`: This is a control to group multiple elements using the same name attribute. Only one from the provided list can be selected.
- `range`: This a way to select a range of numbers.
- `reset`: This resets the form.
- `search`: This is a text input.
- `submit`: This is a button that will submit the form.
- `tel`: This is an input to enter a phone number.
- `text`: This is your basic text input.
- `time`: This is the time without a time zone.
- `url`: This an input to enter a URL. This will do validation as well.
- `week`: This is to enter the week number.

Here is an example of the `text`, `e-mail`, and `tel` inputs:

```
<input type="text" name="name" placeholder="enter email"/>
<input type="email" />
<input type="tel" />
```

button

The syntax of the `button` element is as follows:

```
<button autofocus disabled form formaction formenctype
  formmethod formnovalidate formtarget name type value></button>
```

Attributes

The attributes that are used in the `button` element are as follows:

- `autofocus`: This lets the browser automatically focus on the element that this is an attribute of. This should only be used on one element.
- `disabled`: This states whether or not to disable the element.
- `form`: This is the ID of the form.
- `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget`: These will override the form's value if this is a button or image.
- `name`: This is the name of the button for the form.
- `type`: This changes what the button does. The values are `submit`—which submits the form, `reset`— which resets the form, `button`—which is the default that does nothing.
- `value`: This is the initial value of the element.

Description

The `button` element creates a button that can be clicked on. Changing the `type` attribute will change its behavior.

Here is an example of the `reset` and `submit` buttons:

```
<button type="reset">Reset</button>
<button type="submit">Submit</button>
```

select

The syntax of the `select` element is as follows:

```
<select autofocus disabled form multiple name required
  size ></select>
```

Attributes

The attributes that are used in the `button` element are as follows:

- `autofocus`: This lets the browser automatically focus on this element. This should only be used on one element.
- `disabled`: This states whether or not to disable the element.
- `form`: This is the ID of the form.

- `multiple`: This states whether or not multiple items can be selected.
- `name`: This is the name of the element.
- `required`: This checks whether the option needs a value.
- `size`: This determines the number of rows for the element.

Description

The `button` element is used with the `option` element. Multiple `option` elements can be added to the list to select or choose from. The value of the selected option is used when the form is submitted.

Here is an example:

```
<select name="select">
    <option value="1">One</option>
    <option value="2">Two</option>
</select>
```

See also

You can also refer to the `optgroup` and `option` attributes to find out more about the `button` element.

optgroup

The `optgroup` element is the option group element:

```
<optgroup disabled label></optgroup>
```

Attributes

The attributes that are used in the `optgroup` element are as follows:

- `disabled`: This disables the group
- `label`: This is the heading in the drop-down menu

Description

The `optgroup` element allows you to group options together. The children of this element need to be the `option` elements. They are not selectable and do not have a value.

Here is an example of the `outgroup` element with car makes and models:

```
<select name="cars">
   <optgroup label="Ford">
       <option value="Fiesta">Fiesta</option>
       <option value="Taurus">Taurus</option>
   </optgroup>
    <optgroup label="Honda">
        <option value="Accord">Accord</option>
        <option value="Fit">Fit</option>
    </optgroup>
</select>
```

See also

You can also refer to the `option` and `select` elements to learn more about the `optgroup` element.

option

The syntax of the `option` element is as shown:

```
<option disabled selected value></option>
```

Attributes

The attributes that are used in the `option` element are as follows:

- `disabled`: This states whether or not the element is disabled.
- `selected`: This states whether or not the option is selected. We can set only one option per selected element.
- `value`: This states the value of the `option`.

Description

The `option` elements are the actual items in the `select` element. They can either be the child of a `select` or `optgroup` element.

Here is an example:

```
<select name="select">
    <option value="1">One</option>
    <option value="2">Two</option>
</select>
```

See also

You can also refer to the `select` and `optgroup` elements to find out more about the `option` element.

textarea

The syntax of the `textarea` element is as follows:

```
<textarea autocomplete autofocus cols disabled form
  maxlength minlength name placeholder readonly required
  rows spellcheck wrap></textarea>
```

Attributes

The attributes that are used in the `textarea` element are as follows:

- `autocomplete`: This states whether the browser can autocomplete this input based on previous values or not.
- `autofocus`: This lets the browser automatically focus on this element. This should only be used on one element.
- `cols`: This states the width of the `textarea` element in characters.
- `disabled`: This states whether or not to disable the element.
- `form`: This is the ID of the form.
- `maxlength`: This is the maximum number of characters.
- `minlength`: This is the minimum number of characters.
- `name`: This is the name of the element.
- `placeholder`: This is the text that is displayed in the element when there is no value.
- `readonly`: This makes the element read-only.
- `required`: This states that the element is required and cannot be blank.
- `rows`: This states the number of rows for `textarea`.
- `spellcheck`: This states whether or not the element should have the spelling checked.
- `wrap`: This states how the lines are wrapped.

Description

You will use this when you need more text than just a single line.

Here is an example:

```
<textarea cols="20" rows="10"
  placeholder="Input text here"></textarea>
```

Drawing elements

In previous versions of HTML, if you wanted a graphic or image, you had to create it in another application and use the `img` element to pull it into your document. HTML5 has brought some new elements and features to replace the old elements that allow you to draw your own images in the browser.

canvas

The syntax of the `canvas` element is as follows:

```
<canvas height width></canvas>
```

Attributes

The attributes that are used in the `canvas` element are as follows:

- `height`: This is an attribute to set the height
- `width`: This is an attribute to set the width

Description

The `canvas` element is used for drawing. You can use JavaScript to draw lines, shapes, and images; pull frames from videos; and use WebGL, to name just a few features. The HTML element is just the canvas (aptly named!) that you use to make a drawing. All of the interaction happens in JavaScript.

Here is an example of a small `canvas` element:

```
<canvas height="400" width="400">
    Your browser does not support the canvas element.
</canvas>
```

svg

The `svg` element is the **Scalable Vector Graphics** (**SVG**) element:

```
<svg height viewbox width ></svg>
```

Attributes

The attributes that are used in the `svg` element are as follows:

- `height`: This is the attribute that sets the height.
- `viewbox`: This sets the bounds for the element. It takes four numbers that map to `min-x`, `min-y`, `width`, and `height`.
- `width`: This is the attribute that sets the width.

Description

SVG is not a true HTML element. It is its own specification with many elements and attributes. There are books written entirely about SVG. This element is here because you can now create inline SVG in an HTML document. This gives you a lot of flexibility with two-dimensional drawings that images do not give you.

Here is an example that demonstrates the difference between height, width, and viewport. The viewport takes up the bounds of the element, and height and width give the element its size:

```
<svg xmlns="http://www.w3.org/2000/svg"
  preserveAspectRatio=""
    width="200" height="100" viewBox="0 0 400 200">
    <rect x="0" y="0" width="400" height="200"
      fill="yellow" stroke="black" stroke-width="3" />
</svg>
```

2
HTML Attributes

HTML is built using elements. Each of these elements will have attributes. The attributes can change how the browser renders the element, configures it, and the behavior involving the element.

The focus of the chapter will be entirely on attributes. If you are unsure about which attributes apply to which elements, the previous chapter goes over almost all the elements and the attributes that apply to them.

Global attributes

These are attributes that are available for every HTML element. However, you should note that just because the attribute is available, it does not mean that it will actually do anything.

accesskey

The `accesskey` attribute creates a keyboard shortcut to activate or focus on the element:

```
<element accesskey></element>
```

Description

The `accesskey` attribute allows you to create a keyboard shortcut. This can be a space-delimited list of characters. Most browsers on Windows will use *Alt* + accesskey and most browsers on Mac use *Ctrl* + *Option* + accesskey.

Here is an example using a textbox that can be focused on with the character `q`:

```
<input type="search" name="q" accesskey="q"/>
```

class

The `class` attribute is often used to help group similar elements for CSS selectors:

```
<element class></element>
```

Description

The `class` attribute is one of the most used attributes. The `class` attribute allows CSS to target multiple elements and apply a style to them. In addition to this, many people also use the `class` attribute to help target elements in JavaScript.

Class takes a space-delimited list of class names.

Here is an example that applies the `search-box` class to an element:

```
<input type="search" name="q" class="search-box"/>
```

contenteditable

The `contenteditable` attribute sets the element's content as editable:

```
<element contenteditable></element>
```

Description

The `contenteditable` attribute tells the browser that the user can edit the content in the element. The `contenteditable` attribute should have a value of `true` or `false`, where `true` means that the element is editable.

Here is an example with a `div` element:

```
<div contenteditable="true">
  Click here and edit this sentence!</div>
```

data-*

The `data-*` attribute is the custom attribute for elements:

```
<element data-*></-></element>
```

Description

You can name the data attribute whatever you want as long as the name does not start with XML, does not use any semicolons, and does not have any uppercase letters. The value can be anything.

Here is a list of items with the `data-id` attributes. Note that the attribute name `data-id` is arbitrary. You can use any valid name here:

```
<li data-id="1">First Row</li>
<li data-id="2">Second Row</li>
<li data-id="3">Third Row</li>
```

dir

The `dir` attribute defines the text direction:

```
<element dir></element>
```

Description

The `dir` attribute is the direction attribute. It specifies the text direction. The following are its possible values:

- `auto`: This lets the browser choose the direction automatically
- `ltr`: This lets the browser choose the left to right direction
- `rtl`: The browser chooses the right to left direction

Here is an example for the `ltr` and `rtl` attributes.

```
<div dir="ltr">Left to Right</div>
<div dir="rtl">Right to Left</div>
```

draggable

The `draggable` attribute defines whether the element is draggable:

```
<element draggable-></element>
```

Description

The `draggable` attribute allows the element to be dragged around. Note that most elements require JavaScript as well for this to work fully:

Here is an example:

```
<div draggable="true">You can drag me.</div>
```

hidden

The `hidden` attribute prevents the rendering of the element:

```
<element hidden></element>
```

Description

The `hidden` attribute is used to hide elements. However, a hidden element can be overridden and displayed through CSS or JavaScript. This is a Boolean attribute so including this attribute sets the value to `true` and excluding it sets the value to `false`.

Here is an example:

```
<div hidden>This should not show</div>
```

id

The `id` attribute is a unique identifier of the element:

```
<element id></element>
```

Description

The `id` attribute is a unique identifier of the element. This is used for fragment linking and easily accessing the element in JavaScript and CSS.

Here is an example using a `div` element:

```
<div id="the-first-div">This is the first div.</div>
```

lang

The `lang` attribute defines the language used in the element:

```
<element lang></element>
```

Description

The `lang` attribute sets the language for the element. The acceptable value should be a BCP47 language. There are too many to list here, but you can read the standard at `http://www.ietf.org/rfc/bcp/bcp47.txt`. The language is important for things such as screen readers to use correct pronunciation.

Here is a simple example using English:

```
<div lang="en">The language of this element is English.</div>
```

spellcheck

The `spellcheck` attribute specifies whether spell check can be used:

```
<element spellcheck></element>
```

Description

The `spellcheck` attribute was introduced in HTML5. It will tell the browser whether to spellcheck this element or not. The value should either be `true` or `false`.

Here is an example using textarea:

```
<textarea spellcheck="false">
  Moste fo teh worsd r mispeld.</textarea>
```

style

The `style` attribute is used to set the inline style:

```
<element style></element>
```

Description

You can add CSS styles directly to an element with the `style` attribute. Any style rule that you can use in CSS, you can use here. Remember that this will take precedence over any other styles defined in CSS.

Here is an example that sets the background to red and text to white:

```
<div style="background: #ff0000; color: #ffffff">
  This has inline styles.</div>
```

tabindex

The `tabindex` attribute sets the tab order:

```
<element tabindex></element>
```

Description

The `tabindex` attribute element defines the order in which elements will focus when the *Tab* key is used. There are three different types of values you can use. The first is a negative number. This defines that it is not in the list of elements for tab order. A `zero` value means that the browser should determine the order of this element. This is usually the order in which the elements occur in the document. This is a positive number and it will set the tab order.

The following example demonstrates that you can set `tabindex` in a different order to where the elements are in the document:

```
<input type="text" tabindex="1" />
<input type="text" tabindex="3" />
<input type="text" tabindex="2" />
```

title

The `title` attribute is the text for a tooltip:

```
<element title></element>
```

Description

The `title` attribute gives extra information about the element. Usually, the title is shown as a tooltip for the element. For example, when using an image, this could be the name of the image or a photo credit:

```
<p title="Some extra information.">
  This is a paragraph of text.</p>
```

Miscellaneous

The miscellaneous grouping of attributes will have no hierarchy as they can be used on many different elements.

accept

The `accept` attribute gives the list of types for the server:

```
<element accept></element>
```

Elements

The elements used in the `accept` attribute are `form` and `input`.

Description

The `accept` attribute allows you to suggest the file type that this form or input should accept. You can use `audio/*`, `video/*`, `image/*`, a MIME type, or the extension.

Here is an example looking for PNG files:

```
<input type="file" accept=".png, image/png"/>
```

accept-charset

The `accept-charset` attribute gives the list of support charsets:

```
<element accept-charset></element>
```

Elements

The `form` element is used in the `accept-charset` attribute.

Description

The `accept-charset` attribute sets the charset that the form will accept. UTF-8 is most commonly used as it accepts many characters from many languages.

Here is an example of using the `charset` attribute:

```
<form accept-charset="UTF-8"></form>
```

action

The `action` attribute is where the form is processed, the syntax is as follows:

```
<element action ></element>
```

Elements

The `form` element is used in the `action` element.

Description

The `action` attribute consists of the URL that will process the form's data.

Here is an example:

```
<form action="form-process.php"></form>
```

alt

The `alt` attribute is an alternative text for the element:

```
<element alt ></element>
```

Elements

The elements that are used in the `alt` attribute are `applet`, `area`, `img`, and `input`.

Description

The `alt` attribute is an alternate text in case the element cannot render the code. The text should pass the same information to the user that the image would have.

Some browsers (Chrome being the most commonly used) will not display the text, and you may need to use the `title` attribute. This is not the standardized behavior.

Here is an example using an image:

```
<img alt="Login button." src="non-loading-image.png"/>
```

async

The `async` attribute is used for the asynchronous execution of the script:

```
<element async ></element>
```

Elements

The `script` element is used in the `async` attribute.

Description

The `async` attribute tells the browser to load and execute the script asynchronously. By default, the browser will load scripts synchronously. An asynchronous load will immediately download and parse the script without blocking the rendering of the page.

Here is an example:

```
<script src="application.js" async></script>
```

autocomplete

The `autocomplete` attribute defines whether the element can be autocompleted:

```
<element autocomplete ></element>
```

Elements

The `form` and `input` elements are used in the `autocomplete` attribute.

Description

The `autocomplete` attribute lets the browser know whether or not it can autocomplete the form or input from the previous values. This can have `on` and `off` as the values.

Here is an example:

```
<input type="text" autocomplete="off"
  placeholder="Will not autocomplete"/>
```

autofocus

The `autofocus` attribute defines whether the element would be focused automatically on the elements:

```
<element autofocus ></element>
```

Elements

The `button`, `input`, `select`, and `textarea` elements are used in the `autofocus` attribute.

Description

The `autofocus` attribute will set the focus to the element. This should only be used on one element.

Here is an example of the `autofocus` attribute with a text input:

```
<input type="text" autofocus/>
```

autoplay

The `autoplay` attribute defines whether the audio or video track should play as soon as possible:

```
<element autoplay ></element>
```

Elements

The `audio` and `video` elements are used in the `autoplay` attribute.

Description

The `autoplay` attribute will make the element play as soon as it can, without having to stop to load.

Here is an example with an audio file:

```
<audio autoplay src="audio.mps"></audio>
```

autosave

The `autosave` attribute defines whether the previous values should be saved:

```
<element autosave ></element>
```

Elements

The `input` element is used in the `autosave` attribute.

Description

The `autosave` attribute tells the browser to save values entered into this input. This means that on the next page load, the values will be persisted. It should have a unique name that the browser can associate the saved values to. This attribute may or may not work in some browsers as it is not fully standardized.

Here is an example:

```
<input type="text" autosave="textautosave" />
```

cite

The `cite` attribute has the source of the quote:

```
<element cite ></element>
```

Elements

The `blockquote`, `del`, `ins`, and `q` elements are used in the `cite` attribute.

Description

The `cite` attribute points to the source of a quote by providing a URL.

Here is an example:

```
<blockquote cite=
  "http://en.wikiquote.org/wiki/The_Good,_the_Bad_and_the_Ugly">
    After a meal there's nothing like a good cigar.
</blockquote>
```

cols

The `cols` attribute gives the number of columns:

```
<element cols ></element>
```

Elements

The `textarea` element is used with the `cols` attribute.

Description

The `cols` attribute gives the number of columns in a `textarea` element.

Here is an example of it in use:

```
<textarea cols="30"></textarea>
```

colspan

The `colspan` attribute gives the number of columns a cell should span:

```
<element colspan ></element>
```

Elements

The `td` and `th` elements are used in the `colspan` attribute.

Description

The `colspan` attribute gives the number of columns a table cell should span.

Here is an example using a table element:

```
<table>
    <tr><td colspan="2">1 and 2</td></tr>
    <tr><td>1</td><td>2</td></tr>
</table>
```

datetime

The `datetime` attribute gives the date and time associated with this element:

```
<element datetime ></element>
```

Elements

The `del`, `ins`, and `time` elements are used with the `datetime` attribute.

Description

The `datetime` attribute should be the time that the action implied by the element was taken. This attribute is mainly used with the `del` and `ins` elements to show when the deletion or insertion occurred.

Here is an example using `del`:

```
My name is <del datetime="2014-12-16T23:59:60Z">John</del>Josh.
```

disabled

The `disabled` attribute defines whether the element can be used or not:

```
<element disabled ></element>
```

Elements

The `button`, `fieldset`, `input`, `optgroup`, `option`, `select`, and `textarea` elements are used in the `disabled` attribute.

Description

The `disabled` attribute makes the element unusable. If the element is disabled, it cannot be used. This means that buttons cannot be clicked, text areas and text inputs cannot have text entered, dropdowns cannot be changed, and so on.

Here is an example with a `button` element:

```
<button disabled>This is a disabled button</button>
```

download

The `download` attribute sets a link to download a resource:

```
<element download ></element>
```

Elements

The `a` element is used in the `download` attribute.

Description

The `download` attribute tells the browser to download the resource when clicked on. This means that when the `a` element is clicked, a save dialog will appear with the value of the attribute as the default name.

Here is an example:

```
<a href="example.pdf" download="example.pdf">Save the PDF</a>
```

content

The `content` attribute gives a value to go with the name attribute:

```
<element content ></element>
```

Elements

The `meta` element is used in the content attribute.

Description

The `content` attribute is an attribute for the `meta` tag. It is used as the value of the `name` attribute.

Here is an example:

```
<meta name="example" content="value for example" />
```

controls

The `controls` attribute defines whether the controls should be displayed:

```
<element controls ></element>
```

Elements

The `audio` and `video` elements are used in the `controls` attribute.

Description

The `controls` attribute tells the browser to display controls for a media file. This is a Boolean attribute.

Here is an audio example:

```
<audio controls src="example.mp3"></audio>
```

for

The `for` attribute sets the element this attribute is associated with:

```
<element for ></element>
```

Elements

The `label` element is used with the `for` attribute.

Description

The `for` attribute specifies which form input the label is associated with. This is specified using the ID of the input element. The label will also allow the user to click on it and focus on the input.

Here is an example with a text input:

```
<label for="username">Username</label>
<input type="text" id="username" name="username" />
```

form

The `form` attribute sets the form with which this input is associated:

```
<element form ></element>
```

Elements

The `button`, `fieldset`, `input`, `labellable`, `object`, `output`, `select`, and `textarea` elements are used in the `form` attribute.

Description

The `form` attribute references the form that these controls are in:

```
<form method="get" id="example-form">

</form>
<input type="text" form="example-form" />
```

formaction

The `formaction` attribute sets the form action for the element:

```
<element formaction ></element>
```

Elements

The `button` and `input` elements are used in the `formaction` attribute.

Description

The `formaction` attribute will override the action of the form for this element. This should be a URL. If used on the form element itself, this attribute specifies the target for the form data. If used on an element within the form (for example, button), this overrides the declared value on the form itself.

Here is an example with a button:

```
<form method="get" action="formaction.php">
    <button formaction="buttonaction.php">Press me</button>
</form>
```

height

The `height` attribute sets the height of an element:

```
<element height ></element>
```

Elements

The `canvas`, `embed`, `iframe`, `img`, `input`, `object`, and `video` elements are used in the `height` attribute.

Description

The `height` attribute sets the height of the element. Only the elements listed in the previous section should use this attribute and all other elements should use CSS to set their height.

You may see many HTML documents that use `height` on many elements. This is not valid anymore and CSS should be used to set the height on any other elements.

Here is an example with a canvas:

```
<canvas height="400" width="400"></canvas>
```

href

The `href` attribute gives the URL of the resource:

```
<element href ></element>
```

Elements

The `a`, `base`, and `link` elements are used in the `href` attribute.

Description

The URL for the element is given by the `href` attribute.

Here is an example with an anchor element:

```
<a href="http://www.google.com">Google</a>
```

hreflang

The `hreflang` attribute states the language of the resource:

```
<element hreflang ></element>
```

Elements

The `a` and `link` elements are used in the `hreflang` attribute.

Description

The `hreflang` attribute is the language of the linked document. The acceptable values should be in the BCP47 language. There are too many to list here, but you can read the standard at `http://www.ietf.org/rfc/bcp/bcp47.txt`.

Here is an example:

```
<a href="http://www.google.com" hreflang="en">Google</a>
```

label

The `label` attribute states the title of the track:

```
<element label ></element>
```

Elements

The `track` element is used in the `label` attribute.

Description

The `label` attribute is used with the `track` element to give a title to the track.

Here is an example with subtitles for a video:

```
<video src="sample.mp4">
    <track kind="subtitles" label="English Subtitles"
      src="en.vtt" />
</video>
```

list

The `list` attribute gives the list of options:

```
<element list ></element>
```

Elements

The `input` element is used in the `list` attribute.

Description

The `list` attribute ties to a `datalist` attribute with a list of options for `input`.

This example has a list of fruits for a text input:

```
<input type="text" list="fruit" />
<datalist id="fruit">
    <option>Apple</option>
    <option>Banana</option>
</datalist>
```

loop

The `loop` attribute defines whether the element should loop the media:

```
<element loop ></element>
```

Elements

The `audio` and `video` elements are used in the `loop` attribute.

Description

The `loop` attribute is a Boolean attribute which will play the media on a loop.

Here is an example with an `audio` element:

```
<audio src="example.mp3" loop></audio>
```

max

The `max` attribute defines the maximum value:

```
<element max ></element>
```

Elements

The `input` and `progress` elements are used in the `max` attribute.

Description

The `max` attribute sets the maximum numeric or date-time value allowed.

Here is an example with an input:

```
<input type="number" min="0" max="5" >
```

maxlength

The `maxlength` attribute defines the maximum number of characters:

```
<element maxlength ></element>
```

Elements

The `input` and `textarea` elements are used in the `maxlength` attributes.

Description

The `maxlength` attribute sets the maximum number of characters.

Here is an example with an input:

```
<input type="text" maxlength="5">
```

media

The `media` attribute sets the media for the linked resource:

```
<element media ></element>
```

Elements

The `a`, `area`, `link`, `source`, and `style` elements are used in the `media` attribute.

Description

The `media` attribute specifies which media this resource is for. Usually, this attribute is used with link and the CSS. The standardized values are `screen`, `print`, and `all`. The `screen` value is for displaying on a monitor, the `print` value is when printing, and the `all` value is both. Different browsers do have a few other values, but nothing that works across all.

Here is an example with CSS to create a print stylesheet:

```
<link rel="stylesheet" href="print.css" media="print"/>
```

method

The `method` attribute defines the HTTP method of form:

```
<element method ></element>
```

Elements

The `form` element is used in the `method` attribute.

Description

The `method` attribute sets the HTTP method of the form. The two values are `GET`, which is the default, and `POST`.

Here is an example of a form that submits using the `POST` HTTP method:

```
<form method="post" action="formaction.php"></form>
```

min

The `min` attribute defines the minimum value:

```
<element min ></element>
```

Elements

The `input` element is used in the `min` attribute.

Description

The `min` attribute is the opposite of `max`. It sets the minimum value for an input.

Here is an example:

```
<input type="number" min="2">
```

multiple

The `multiple` attribute defines whether multiple values can be selected:

```
<element multiple ></element>
```

Elements

The `select` element is used in the `multiple` attribute.

Description

The `multiple` attribute allows you to select multiple values. This is a Boolean attribute.

Here is an example:

```
<select multiple>
    <option>First</option>
    <option>Second</option>
</select>
```

name

The `name` attribute is the name of the element:

```
<element name ></element>
```

Elements

The `button`, `form`, `fieldset`, `iframe`, `input`, `object`, `select`, `textarea`, and `meta` elements are used in the `name` attribute.

Description

The `name` attribute names an element. This allows you to get the value of the element in a submitted form.

Here is an example with an input:

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

novalidate

The `novalidate` attribute defines whether the validation should be skipped:

```
<element novalidate ></element>
```

Elements

The `form` element is used in the `novalidate` attribute.

Description

The `novalidate` attribute sets the form to not validate when submitted. The browser will validate the input without adding any client-side code. This is a Boolean attribute.

Here is an example:

```
<form method="post" action="formaction.php" novalidate></form>
```

pattern

The `pattern` attribute defines a regular expression:

```
<element pattern ></element>
```

Elements

The `input` element is used in the `pattern` attribute.

Description

You can use a regular expression in this attribute to validate the input.

Here is an example that will only be valid with numbers:

```
<input pattern="[0-9].+" type="text" />
```

placeholder

The `placeholder` attribute gives a hint for the user in the element:

```
<element placeholder ></element>
```

Elements

The `input` and `textarea` elements are used in the `placeholder` attribute.

Description

When the element has not been interacted with (no value and is not in focus), it will display the text in this attribute. The value will disappear once the element is interacted with.

Here is an example with `input`:

```
<input type="text" name="username"
  placeholder="Please enter username"/>
```

poster

The `poster` attribute gives an image for a video:

```
<element poster ></element>
```

Elements

The `video` element is used in the `poster` attribute.

Description

The `poster` attribute should point to an image that will be the poster (or placeholder) for a video element until the video is loaded.

Here is an example:

```
<video src="video-about-dogs.mp4"
  poster="images/image-of-dog.png"></video>
```

readonly

The `readonly` attribute defines whether the element is editable:

```
<element readonly ></element>
```

Elements

The `input` and `textarea` elements are used in the `readonly` attribute.

Description

The `readonly` attribute makes the element readonly or not editable. This is a Boolean attribute.

Here is an example with a text input:

```
<input type="text" readonly />
```

rel

The `rel` attribute defines the relationship of the element:

```
<element rel ></element>
```

Elements

The `a` and `link` elements are used in the `rel` attribute.

Description

The `rel` attribute is the relationship between the linked resource and the document. Usually, you will see this used with the `link` element and CSS or with the `a` element and the value of `nofollow`.

Here is a CSS example:

```
<link rel="stylesheet" href="style.css"
  type="text/css" media="screen" />
```

required

The `required` attribute defines whether the element is required when submitting a form:

```
<element required ></element>
```

Elements

The `input`, `select`, and `textarea` elements are used in the `required` attribute.

Description

The `required` attribute makes the element required in the form. The form will not get submitted until the element is filled out. This is a Boolean attribute.

Here is an example with a text input:

```
<input required type="text" />
```

reversed

The `reversed` attribute changes the list order display:

```
<element reversed ></element>
```

Elements

The `ol` element is used in the `reversed` attribute.

Description

The `reversed` attribute is only an attribute of an ordered list, and it will render the list in the reverse order. The items are not in reverse, but rather the numbered list indicators are. This is a Boolean attribute.

Here is an example:

```
<ol reversed>
    <li>1</li>
    <li>2</li>
    <li>3</li>
</ol>
```

rows

The `rows` attribute sets rows in a text area:

```
<element rows ></element>
```

Elements

The `textarea` element is used in the `rows` attribute.

Description

The `rows` attribute sets the number of rows in `textarea`.

Here is an example:

```
<textarea rows="30"></textarea>
```

rowspan

The `rowspan` attribute sets the number of rows that a cell will span:

```
<element rowspan ></element>
```

Elements

The `td` and `th` elements are used in the `rowspan` attribute.

Description

The `rowspan` attribute is used on the table cell elements. It will make the cell span the number of rows in the value.

Here is an example:

```
<table>
    <tr><td rowspan="2">Pets</td><td>Dogs</td></tr>
    <tr><td>Cats</td></tr>
</table>
```

scope

The `scope` attribute defines the cell with which the element is associated:

```
<element scope ></element>
```

Elements

The `td` and `th` elements are used in the `scope` attribute.

Description

The `scope` attribute indicates what the cell is associated with. For example, this could be `row` or `col`. This is a great benefit for accessibility. It helps to convey the relationship of the rows and columns in the table when a screen reader is used.

Here is an example with a table cell:

```
<table>
    <tr><th>Name</th><th>Age</th></tr>
    <tr><td scope="row">Gizmo</td><td>4</td></tr>
</table>
```

selected

The `selected` attribute sets the default selection:

```
<element selected ></element>
```

Elements

The `option` element is used in the `selected` attribute.

Description

The `selected` attribute will set the option to be selected when the page loads. This should only be set on one `option` element per `select` element. This is a Boolean attribute.

Here is an example:

```
<select>
    <option>Cats</option>
    <option selected>Dogs</option>
</select>
```

size

The `size` attribute sets the width of element:

```
<element size ></element>
```

Elements

The `input` and `select` elements are used in the `size` attribute.

Description

The `size` attribute determines the width of the element unless the element is an input and text or password. It will then determine the number of characters the element can display. This is specified as the number of characters in the integer form. The default for this is `20`.

Here is an example with a text input:

```
<input type="text" size="100" />
```

src

The `src` attribute gives the URL for the element:

```
<element src ></element>
```

Elements

The `audio`, `embed`, `iframe`, `img`, `input`, `script`, `source`, `track`, and `video` elements are used in the `src` attribute.

Description

The `src` attribute gives the URL of the resource for the element.

Here is an example with an `image` element:

```
<img src="dogs.png" />
```

start

The `start` attribute sets the starting number:

```
<element start ></element>
```

Elements

The `ol` element is used in the `start` attribute.

Description

The `start` attribute will change the starting number for an ordered list.

Here is an example:

```
<ol start="10">
    <li>1</li>
    <li>2</li>
</ol>
```

step

The `step` attribute determines the jump between each number:

```
<element step ></element>
```

Elements

The `input` element is used in the `step` attribute.

Description

The `step` attribute is used with an `input` element and the `type` attribute of number or date-time. It determines how far to step for each increment.

Here is an example with a number input. You will only be able to increment by `5` four times before reaching the max:

```
<input type="number" min="0" max="20" step="5" />
```

type

The `type` attribute defines the type of the element:

```
<element type ></element>
```

Elements

The `button`, `input`, `embed`, `object`, `script`, `source`, and `style` elements are used in the `type` attribute.

Description

The `type` attribute is one of the most complex attributes. It can completely change the look and behavior of an element. For example, `input`.

Here is a list for each element:

- `button`: Following are the values of the `button` attribute:
    - `button`: This is the default value
    - `reset`: This resets the form
    - `submit`: This submits the form
- `input`: Please view the input element section from the previous chapter.
- `embed`: This will be the MIME type of the embedded resource.
- `object`: This will be the MIME type of the object.
- `script`: This will be the MIME type. Usually `text/javascript` are used.
- `source`: This will be the MIME type.
- `style`: This will be MIME type. Usually `text/css` are used.

Here is an example with an input:

```
<input type="password" />
```

value

The `value` attribute sets the value of the element:

```
<element value ></element>
```

Elements

The `button`, `input`, `li`, `option`, `progress`, and `param` elements are used in the `value` attribute.

Description

The `value` attribute will set the value of the element at page load.

Here is an example with a text input:

```
<input type="text" value="Hey!"/>
```

width

The `width` attribute sets the width of the element:

```
<element width ></element>
```

Elements

The `canvas`, `embed`, `iframe`, `img`, `input`, `object`, and `video` elements are used in the `width` attribute.

Description

The `width` attribute sets the width of the element.

You may see many HTML documents that use width on many elements. This is not valid any more; CSS should be used to set the width on any other elements.

Here is an example with a `canvas` element:

```
<canvas width="200" height="200"></canvas>
```

wrap

The `wrap` attribute sets how the text is wrapped:

```
<element wrap ></element>
```

Elements

The `textarea` element is used in the `wrap` attribute.

Description

The `wrap` attribute decides whether or not text can be wrapped in `textarea`. The values can be hard or soft. A hard value will insert line breaks into the text. It must also be used with the `cols` attribute, so it knows where the end of the line is. A soft value will not insert any line breaks.

Here is an example:

```
<textarea cols="10" wrap="hard"></textarea>
```