Chapter 4: HTML Elements

The HTML standard (https://html.spec.whatwg.org/) defines legal elements and gives them semantics. Browsers and other applications that display HTML use this standard and definitions to display HTML documents. Elements are roughly divided into classes based on their use, with each element being in one or more classes:

- Metadata
- Flow
- Sectioning – chapters and areas
- Heading – headings
- Phrasing
- Embedded
- Interactive

For each element, the standard defines quite precisely how the element behaves and how it should be implemented. We as *authors* 1 (#footnote:1) we need to look at the definition of how to use, preferably:

- Where the element can be used, " contexts in which this element can be used "
- What the element can contain, " content model "
- What attribute the element has, " content attributes "
- What to consider regarding *accessibility issues* , " accessibility considerations "

1 (#footnote-reference:1) The standard is intended for both website authors—those who write HTML—and browser authors—who interpret and display HTML.

Following general definitions is a textual description of the use of elements, along with examples of usage.

When writing HTML, these are the elements we consider semantically. To help with your search, it's a good idea to use the W3C definitions (https://html.spec.whatwg.org/multipage/dom.html#kinds-of-content) , MDN's "HTML element reference (https://developer.mozilla.org/en-US/docs/Web/HTML/Element) ," or W3C's "The Elements of HTML (https://w3c.github.io/elements-of-html/) . "

Based on the meaning of these elements, the browser sets a basic appearance, e.g. headings are larger, paragraphs have a bottom margin, and links are blue with an underline. We then change this appearance with CSS , not by selecting different (and potentially) wrong elements!)

What follows is a review of the main elements of HTML.

Metadata 4.1

Metadata is data that defines how the content of a web page should be displayed, links other data to it (such as CSS files), or defines the position of a page in a group of pages. This metadata is not displayed directly to the user in the browser window and is most often defined in `<head>`.

- `<style>` we use to embed CSS definitions *directly into* the document without having it in a separate CSS file.
- `<title>` is the element that should always be defined to set the title of a document.

`<link>` 4.1.1

To form a connection with other data [2] (#footnote:2) we use `<link>` (https://html.spec.whatwg.org/multipage/semantics.html#the-link-element) . If `rel`the attribute is set, then we are limited `<link>`to `<head>`, usually with `href`which refers to a document, e.g.

2 (#footnote-reference:2) this link i not *a hyperlink* , but a link, for example, to CSS that defines appearance, or a JavaScript file tha defines functionality.

- `rel="stylesheet"`defines a reference to a CSS file to be retrieved and displayed.
- `rel="next"`defines which document is next in the queue, browsers can use this as a suggestion to fetch that document before a user requests it, thus speeding up the process.
- `rel="prefetch"`suggests to the browser to retrieve data while HTML parsing is taking place.
- `rel="preload"`defines that the marked data *must* be retrieved as soon as possible, the type of the data is defined by `as`the attribute. With this we can, for example, instruct the browser to retrieve font [3] which is used as soon as possible and thus make it appear sooner rather than later (otherwise the font would be fetched after CSS parsing.)

3 we see how the font is used in CSS walkthrough.

You can add a file to the root of a website called `favicon.ico`or `<link rel="shortcut icon" href="/icon.png">`to get this little icon that usually sits on the left side of the browser bar. The reason for this can be traced back to 1999 when this functionality was half-sneaked into Internet Explorer 5 (http://thehistoryoftheweb.com/how-we-got-the-favicon/) .

`<meta>` 4.1.2

For general metadata we use `<meta>` (https://html.spec.whatwg.org/multipage/semantics.html#the-meta-element) . We have seen how it is used to define a character set with a special `charset`attribute. It can also be used to define other metadata, in which case we define both `name`and `content`attribute. For example, for `author`which defines the author of a page, `description`which defines a general description, or `generator`which defines the tool that created the page.

`<script>` 4.1.3

To reference JavaScript, we use `<script>` (https://html.spec.whatwg.org/multipage/scripting.html#the-script-element) either a <script> element that contains code or a reference to a JavaScript document that contains our code. If we have text inside an <script> element, that code will be executed when the parser comes across it. If `src`the <script> attribute is defined, the referenced document will be fetched, parsed, and interpreted (the code executed) before proceeding, because JavaScript code could output HTML. This is undesirable, but possible, so the browser will have to wait for HTML parsing. For this reason, a <script> tag is `<script>`often placed before the <script> element `<body>`, so the browser can display data to the user before the JavaScript is interpreted. For historical reasons, we cannot self-close `<script>`an element; we must always close it with an end tag.

If a browser does not support JavaScript, or if JavaScript functionality has been disabled, the element can be used `<noscript>` (https://html.spec.whatwg.org/multipage/scripting.html#the-noscript-element) . Its content is displayed *only* if JavaScript functionality is disabled.

Example 4.1.4

An example that utilizes all the metadata mentioned:

```
<!doctype html >

< html  lang = "en" >

  <head>
```

```
    < meta  charset = "utf-8" >

    < meta  name = "author"  content = "NN" >

    < meta  name = "description"  content = "Metadata example" >

    < link  rel = "stylesheet"  href = "styles.css" >

    < link  rel = "preload"  href = "font.woff2"  as = "font" >

    < link  rel = "shortcut icon"  href = "icon.png" >

    < title > Hello world </ title >

  </head>

  <body>

    < p > Hi! </ p >

    < noscript > JavaScript functionality disabled </ noscript >

    < script  src = "scripts.js" > </ script >

  </body>

</html>
```

Flow 4.2

Most elements we use in the body of a web page, `<body>`, are *flow* elements, many of which can also be *palpable* content: they contain something (are not empty) that is not hidden. Some elements fall into only these categories and no others:

- `<address>`, defines information on how to contact the author of a content.
- `<blockquote>`, identifies a section in a document that is cited from another location.
- `<footer>` defines content about the element it is contained in, specifying, for example, the author, related content, or when the content was last updated.
- `<header>` defines a group of materials that introduce or provide guidance on a topic.
- `<main>` defines the main content of a website, this element should only appear once per web page.

- `<p>`, paragraph, in most cases defines text and elements that fall under the wording .
- `<pre>`, *preformatted* , text within `<pre>`is unstyled by the browser and maintains spaces and line spacing.
- Tables and lists .

`<figure>` 4.2.1

When we have a self-contained area within a document that we will want to reference from other places (possibly with explanatory text), we can use `<figure>` (https://html.spec.whatwg.org/multipage/grouping-content.html#the-figure-element) . This could be a diagram, a graph, or program code.

If explanatory text is included, it is defined within the first `<figcaption>`element. The referenced content is placed within the element.

`<dialog>` 4.2.2

`<dialog>` (https://html.spec.whatwg.org/multipage/interactive-elements.html#the-dialog-element) is used to mark a part of a web page that requires or requires user interaction. This could be entering information, selecting an action, or responding to a request. Often this information is displayed in *a mode window* [6] (#footnote:6) (modal window) where the user must take a position on what is displayed. The standard defines an *API* so that general display can be used (mainly opening and closing the "window"), but as of 2021, this functionality is not implemented in all browsers and therefore not usable except with reservations / caveats.

6 (#footnote-reference:6) here bette Icelandic is needed.

`<div>` 4.2.3

Another element that falls under the category of clear content is `<div>`. This element has no real meaning, its meaning is defined by what it contains. Often we use it `<div>`when we don't think about what and what kind of content we are marking, often there are other elements that are more suitable.

„ Authors are strongly encouraged to view the `div`element as an element of last resort, for when no other element is suitable. Use of more appropriate elements instead of the `div`element leads to better accessibility for readers and easier maintainability for authors. “

— Grouping content: The div element

This doesn't mean we never use it though `<div>`! When we start working on adding a layout to our websites, it often happens that we need an extra element, and in many cases it is `<div>`an element.

Example of a flow element 4.2.4

```
<!doctype html >

< html  lang = "en" >

  <head>

    < meta  charset = "utf-8" >

    < title > Flow example </ title >

  </head>

  <body>

    <header>

      < h1 > DocType example </ h1 >

    </header>

    < main >

      < p > In example 1 we see how the smallest HTML document is defined. </ p >

      < figure >

        < before >

          < !doctype html>

          < html lang="en">

            <head>

              < meta charset="utf-8">

              < title>Hello world < /title>

            < /head>

            <body> ... </body>

          </html>

        </pre>

        < figcaption > Example 1: The smallest HTML document </ figcaption >

      </ figure >

      < p > How about having a memex? </ p >
```