2
HTML

In this chapter, we will walk you through the basics of HTML. In *Chapter 1*, *The World Wide Web*, we already covered what the letters HTML mean, where the language comes from, and what it is used for: to create the content part of your web pages. We already know that this content is placed in between tags: `<tag>` to open and `</tag>` to close.

It would be beyond the scope of this book to provide a complete reference to all HTML tags, or elements (we will use these words interchangeably), and all of their attributes. There are some good references listed in the bibliography and of course there are some cool online references. I personally like `w3schools.com` but, if you don't, simply Google "HTML" followed by a tag name you would like to know more about and you will find some great alternate sources.

We will therefore only describe the most commonly used HTML tags in this chapter, grouped by the role they play in the document. For example, all the tags that can be used in a table are grouped under the heading `table`.

HTML versions

Since its creation, there have of course been several different versions and flavors of HTML. The most notable are HTML4, XHTML, and HTML5:

- **HTML4**: This is the last of a series of versions of HTML and is what most people will implicitly refer to when they talk about HTML.
- **XHTML**: This is a different definition of HTML and an attempt to make HTML a member of the XML family, giving it more strict rules. An advantage is that it would be easier to use tools and languages that are intended to manipulate and translate XML documents. However, interest in maintaining that standard seems to have faded.

- **HTML5**: This is the newest kid on the proverbial HTML block. A lot of books have been published about it and, if you have read one of them, you will have discovered that HTML5 is more than just a new version of the markup language. Granted, it comes with quite a few new tags, such as the `<nav>` or `<section>` tags. HTML5 also features the use of custom data attributes such as `data-whateveryouchoose` that you can use in your document. Later on you can manipulate these using JavaScript. It is a way to pass data along inside an element; hence the name chosen: `data-*`.

Did I say JavaScript? All the other new features in HTML5 are actually JavaScript APIs like HTML5 Canvas. Canvas lets you draw things on your web page, pie charts for example. Exciting as these APIs may be, they are beyond the scope of this chapter.

Semantic and presentational HTML

The approach we are taking in this chapter and in the first part of the book overall is to only use HTML elements and attributes that are covered by all three standards. In practice, this means we will not use any HTML4 attributes that disappeared in HTML5 and will not use any HTML5 elements or attributes that did not exist in HTML4.

On the other hand, we do not want to discourage the use of new things, so we will list HTML5-specific elements in a separate list. We will also use the new elements in the second section of the book where we introduce a cool CSS/JavaScript framework.

One could easily divide HTML elements into two groups. The first group consists of elements that refer to parts of a document: headers, paragraphs, tables, forms, lists, and so on. (`<h1>`, `<h2>`, `<p>`, `<table>`, `<ul>`). We call this semantic HTML as they refer to the names of things; they describe what they are.

Another group contains the elements used to indicate how things look: how they are aligned, which font is used, if it is in bold or italics, and so on (`<center>`, `<font>`, `<b>`, `<i>`), and we could call them presentational HTML. The same is true for HTML attributes. `class="green"` or `id="chapter"` would be semantic, while `width="150px"` or `valign="top"` would be presentational.

It is the recommendation of the W3C to use CSS for presentational things, and we follow that recommendation. This way will also avoid you learning a bunch of new things, only to later find out that they are no longer used, as most HTML4 elements and attributes that are no longer available in HTML5 happen to be presentational. The word you will find online to indicate that something is no longer used is *deprecated*.

When I first ran into this word I misread it as depreciated. That word might have been a better choice. Either way, if elements and attributes are labeled as such, avoid using them.

As a consequence, we are not going to show you pretty examples of HTML documents in this chapter, as the part that will make it pretty, CSS, will have to wait for a while until we get to *Chapter 3*, *CSS*.

The structure and syntax of an HTML document

An HTML document is a text file with a name ending in `.html`, for example, `hello.html`. A modern, minimal HTML document looks like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<title>Hello, world</title>
</head>
<body>
<h1>Hello, World</h1>
</body>
<html>
```

Doctype

The first line specifies the document type (`<!DOCTYPE html>`). Today, this can be as simple as the word `html` telling a browser that this file is to be interpreted as an HTML5 document. Documents written to the older specifications contain the name of that spec followed by a path to a **Document Type Definition** (**DTD**) file that can be used to examine the document for conformance. Things are a lot more flexible these days.

<html>

This is the root of the document. Everything in the remainder of the document will be inside this `html` tag. What is inside the `html` tag consists of two sections, the `head` and the `body`.

<head>

The head section contains one or more `<meta>` tags. The one in our earlier example specifies that the encoding of the text part of the document has to be in **UTF-8**.

This section is also where the `<title>` tag lives. The text inside this element is the text that will be displayed at the top of the browser window and is looked at heavily by search engines. So it is important to always include a `title` tag and for its contents to be correct and meaningful.

Furthermore, the `<head>` section is used to include more information that the browser will have to read before the `body` part of the document is loaded. Your most typical example will be the paths to the **CSS stylesheets** that are used for the document. We will have many examples in this book.

<body>

Inside the `body` tag is the core content of our document. As a consequence, if there are certain style elements that you want to be used in the entire document, you will be able to do that by simply styling the `<body>` tag. We will remind you about that later. Of course, we first have to learn what we can put inside the body.

Syntax for tags or elements inside the document

The syntax of an HTML tag is very simple:

```
<tag attribute1="value1" attribute2="value2">text</tag>,
```

This is followed by the `next` tag. You can place everything on a single line or every pair of tags on a separate line for readability as new lines and spaces in between tag pairs are ignored.

Inside the `text` portion, spaces are not ignored, but multiple spaces in a row are reduced to one. So if you want to insert more spaces, you will have to use a different method (See *HTML entities* later in this chapter).

For elements that have no content, there is a shorthand notation. We can use:

```
<tag/>
```

We use that instead of:

```
<tag></tag>
```

In our example, the shorthand notation is used for:

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

The `class` attribute can have multiple values, in which case it would be written like:

```
<tag class="value1 value2">text<?tag>
```

It is not written like this:

```
<tag class="value1" class="value2">text<tag>
```

This last line demonstrates a common oversight when a `class` attribute is added without realizing that a class attribute was already present. In this case, the second one will be ignored. The browser will also ignore all elements and attributes that are not recognized as HTML tags.

Unlike compilers for old school programming languages, you will never see an error message when you mistype something. Things will simply not look right or you may even get a blank screen. This is why it is extremely productive to use an HTML editor or other tool that recognizes tags and attributes as valid HTML ones, preferably tools that display them in color.

HTML comments

Anywhere inside HTML code, you can insert a comment: a reminder to yourself , for posterity, or (probably more important) for others in your team who need to share your code. The syntax is very simple. Anything that is inside an HTML block can be commented by putting `<!--` in the front of it and `-->` after it:

We recommend strongly inserting more comments rather than less. Applying comments is also useful when someone asks you to remove something from the website and you have this feeling that it might come back. Because if you remove it: gone is gone!

On the other hand, every line of HTML comment adds a line to your file and also makes it visible to the world, as every browser has an option to look at the source code. So once you start using a server-side language such as PHP, which you will learn in a few chapters, it is better to place your comments inside that code. You will discover that the syntax for comments in CSS and PHP is different.

As promised, we will now describe the most important HTML tags and attributes, divided in functionality groups. We will start with what started it all: **links**.

Links

In all likelihood, the first web page that was ever created contained a link to the second ever web page. To place a link on a page, we use the anchor tag `<a>`.

The <a> tag and attributes

If we simply place some text inside an `<a>` tag, nothing will really happen when you click on it, unless you program the event in JavaScript. However, you can tell from the way it looks on the screen that the intent is for it to be a link. By default, the content of an `<a>` tag is rendered in the (by now probably notorious) underlined and blue style.

```
<a>Click here</a>
```

The href attribute

To make the link work, you need to use the **href** or **hypertext reference** attribute. You can link to another web page, external or local, a document or image, or another section of the current page. Here are some examples:

```
<a href="http://www.packtpub.com">Visit our website</a>
<a href="index.html">Home</a>
<a href="pdfs/manual.pdf">Click here to download the manual</a>
<a href="#top">Go back to top </a>
```

The first three examples should be self-explanatory. There is a complete URL, a single file name, `index.html`, and a relative path to a PDF file. Absolute pathnames are supported but their use is not recommended. The last example requires more explanation. Did you notice the sharp sign?

The <a> name attribute

The `name` attribute when used in conjunction with the `<a>` tag can be used to name a particular spot on the page. That name can then be used elsewhere on the page in a link.

So, you could put this somewhere near the top of your page:

```
<a name="top"></a>   <!-- note that there does not have to be any
content  -->
```

A link somewhere else on the page, using the same name, but preceded with a `#` sign, will take us back there:

```
<a href="#top">Back to top </a>
```

The <a> target attribute

When a user clicks on a link and arrives at a new page, they sometimes want to go back to where they came from. Some devices and most browsers feature a **back** and even **forward** button a visitor can click on only to discover that the browser does not always take them back to the page they expect. Or the button may not have any effect at all.

In the second half of the book, we will spend an entire chapter on this topic and the notion of what a **previous** page should really be. For now, you can help your cause and your visitor by adding the `target` attribute to your anchor element. It allows you to determine whether or not the `target` page (hence the name of the attribute) will open in a new browser window or not. There are four options:

- **target="_blank"**: This page opens up in a new window or tab
- **target="_self"**: This page opens in the same window it was clicked in; this is the default but also sometimes means that you created a point of no return
- **target="_top"**: This page opens in the full window size of the browser
- **target="_parent"**: This page opens in the parent window

Classic document elements

This section lists a few HTML elements that will look familiar to users of word processors or desktop publishing programs.

<h1>, <h2>, <h3>, … <h6> – headings

These are headings. The smaller the number, the larger the font size the browser will render the heading in.

<p> – paragraph

This is the paragraph tag. Browsers automatically add some space (margin) before and after each `<p>` element. The margins can be modified with CSS (with the margin properties).

– span

The `span` tag by itself has no visual effect but it is extremely useful when you need to style just a portion of text.

You can use it like this:

```
<h3>Example</h3>
<p>This is a paragraph with one <span
class="blue">blue</span>word</span>
```

Lists

In almost every document, you will find the need to sum up a number of items in a list. In HTML you have the choice between an unordered list (think bullets) and an ordered list (think numbers). The HTML elements for these lists are `<ul>` and `<ol>`.

This example produces a list of colors:

```
<h2>Colors</h2>
<ul>
<li>red</li>
<li>green</li>
<li>blue</li>
</ul>
```

This will generate a list of colors, each preceded by a (round) bullet. Replacing `<ul>/</ul> by <ol>/</ol>` will give you a numbered list. Attributes existed to specify the shape of the bullet but these are long gone. Bullet styles are specified in CSS these days. You can even use an image file for the bullet.

A third list element that is worth looking into is `<dl>` or data list.

Images

It is hard to imagine a website without images. Most people assume that adding a picture to a site is easy, that it may take a little bit of Photoshopping and that's it. This is actually not true, but it is all manageable. Being a photographer myself, I was disappointed to discover on my first time experimenting with HTML that putting text right next to a picture on a web page was painful. That was because I did not know enough CSS at the time.

There is actually only one HTML element needed to deal with images: the `<img>` tag.

<img> element and attributes

A typical piece of HTML containing an image would be:

```
<img  src="images/lupine.jog" alt="lupine" />
```

An `img` tag will never have any content inside so we always use the shorthand notation. The two attributes that are always present are `src` and `alt`. The value of the `alt` attribute is a text that will be displayed when the image file cannot be found or when device is used that cannot display images. The `src` attribute contains the path to the image file. An image file can be in one of many different formats: `.jpeg`, `.gif`, `.png`, `.tiff`, etc.

When no information is given about the actual size of the part of the screen that we want to use to display the image, it will be shown at its actual size, so beware of large image files.

Image width and height

There are two attributes you can use for this: **width** and **height**. This will cause the browser to render the image at the size you specify, but it is far better to not use these attributes at and specify the width and height in CSS. So give your `<img>` tag a class or an `id` tag to do so.

You will later learn that you even have the opportunity to specify different image sizes for different screen sizes when we are discussing **responsive** designs.

Either way, once you know what the largest ever size of the image that is going to be used is, create a version of your image file of exactly those dimensions to use on your site. If the original was larger, you will not force the visitor to download a large file that they do not need. If the original was smaller, create a quality image file at the larger size, so it will look good, rather than you relying on how the browser will extrapolate the image.

Input forms

You have all seen them and used them and now you are going to create them: registration forms, order forms—in short: **forms**. What all forms have in common is that the user will enter, or input, some information. Next, that input is validated—for example, to verify that an e-mail address is actually in the correct format—and then it is processed one way or another.

The form will, of course, be written in HTML and CSS. Validation can happen on the client side before it is processed, in JavaScript, and on the server side while it is processed. The processing is, in most cases, done in PHP and the result stored in some kind of database, such as MySQL or MongoDB, or a non-database, such as a flat file, an XML file, or an Excel spreadsheet. For now, let's focus on the creation of the form itself.

Form elements

The elements we will discuss here to be used in forms are : `<form>`, `<label>`, `<input>`, `<textarea>`, `<button>`, `<select>`, and `<option>`. We will treat `<select>` separately.

This is an example of a typical portion of HTML describing a form:

```
<form id="myform" action"processing.php" method="post"
class="formclass">
<label for="title">Title</label>
<input type="radio" value="Ms" id="title" "name="title"
class="classic">Ms.</input>
<input type="radio" value="Mr" id="title" "name="title"
class="classic">Mr</input>
<label for="first">First Name</label>
<input type="text" name="first" id="first" class="classic" />
<label for="last">Last Name </label>
<input type="text" name="last" id="last" class="classic" />
```

```
<label for="email">email</label>
<input type="text" name="email" id="email" class="classic" />
<button type="submit">Register</button>
</form>
```

Form attributes

Notice the `action` and `method` attributes for the `form` tag. They indicate the name of the program that will be used to process the data and the method used to do so. We will explain this in great detail in *Chapter 5*, *PHP*.

The label attribute

The `label` element is a useful tag to label the `input` elements. The `for` attribute ties a `label` tag to an `input` tag.

Input attributes

The `input` element is the most versatile element to be used in a form. It is used to let the user give input, either by typing some text or by checking off a checkbox or radio button.

There are several types, specified by the `type` attribute:

Attribute	Description
`type="text"`	This is the default so there is no need to specify this attribute: this is for text.
`type="hidden"`	This one does not show, but it is extremely useful to pass values.
`type="radio"`	This creates a radio button: only one can be selected.
`type="checkbox"`	This creates a checkbox: multiple checkboxes can be selected.
`type="password"`	This is like text but the inputted characters are not shown.
`type="button"`	This creates a button. You can also create buttons using the `<button>` tag.
`type="submit"`	This creates a submit button. This means the form will be send to the server. You can also create a submit button using the `<button>` element and its `type="submit"` attribute.
`type="file"`	This creates a file upload dialog with a `Choose file` button. When you use the `multiple` attribute and the browser supports it, you can select multiple files.

The name attribute

Every input element that is going to be processed once your visitor has done something with it, needs to have a name. That name will end up being used to create a variable name on the server side when the form is processed. `Radio button` input elements that belong together should be given the same name.

In the case of the `checkbox` type input, you should not only use the same name for every checkbox input element, you also want to use square brackets behind the name. The result of this is that you will have access to an array of all the checked elements on the processing side, which will become extremely handy.

The value attribute

This one is used to assign default values to input elements or to assign values that were previously used in the form and were since stored in a database, as would be the case in any kind of "This was your choice, would you like to change anything?" situation.

The checked attribute

Use this when a `radio button` or `checkbox` needs to be checked by default.

The readonly attribute

If you specify `readonly`, the visitor will not be able to enter any input.

Textarea

When input is expected in a form that is longer than just a few words, you can use the `textarea` element to display an input box. You can specify the size of the box in rows and columns by using the `rows` and `cols` attributes. Here is an example:

```
<textarea row="4" cols="50" id="mytextbox">
</textarea>
```

Drop-down lists

Often we need to have visitors make a choice from a list. For this purpose, we can use the `<select>` element in combination with the `<option>` tag for every individual choice. The text portion of the `<option>` element is what will be displayed, and the value of the `value` attribute is what will be passed on for processing. It is very useful to make the first option the one that is not to be chosen, as in the following example:

```
<select name="colorlist" id="colorlist">
<option value="0">Choose a color</option>
<option value="red">Red</option>
<option value="blue">Blue</option>
<option value="green">Green</option>
<option disabled value="orange">Orange</option>
</select>
```

The disabled attribute

Both the `<select>` and the `<option>` support **disabled** in case you want to enable an option (or the entire drop-down list) to be displayed but not selectable.

The selected attribute

If you want to preselect one of the choices, use the **selected** attribute of `<option>`.

Tables

Tables are heavily used in HTML documents. If you want to present data in a grid of columns and rows, like in a spreadsheet, you may want to create a table. A cell in a table can not only contain numbers or words, it can even contain an image or ... another table.

By default, the browser will make a judgment call as to how wide each column will have to be, depending on the width of the cell contents and how much room there is for the table overall.

Table elements

The following HTML elements are used to create tables:

<table>

This is the main tag to create a table. Every table begins with a `<table>` tag and ends with a `</table>` tag.

<thead> <tbody>

These (optional) elements allow you to separate (and subsequently style) the header part and body part of a table. Like in spreadsheets, the header is used for the descriptions of what goes in the table rows, and the body for the actual content. You can use more than one `<tbody>` section to better organize (and style) your table.

<tr>

No rows? No table. The `<tr>` or **table row** is the element you are going to use for all the rows in your table, both the header and body section.

<th> <td>

These are the elements for your table cells. The `<th>` tags are used for your labels in the table header and the `<td>` (table data) for your content cells in the table body.

Table attributes

Some of the table elements have attributes that are unique to tables. We will discuss them here.

colspan (td)

A table that consists of *x* rows and *y* columns will of course contain *x times y* cells. With the `colspan` attribute, you can specify that, for a given cell, you want to span it across a number of columns. The following line will span this table cell across three columns:

```
<td colspan="3">Verylongname</td>
```

rowspan (td)

This is the equivalent of `colspan` but for rows. With this attribute, you can specify that you want a table cell to be higher than just a single row.

Let's take a look at a table example:

```
<table>
<thead>
<tr><th>First</th><th>Last</th><th>Organization</th><th>Phone</th></
tr>
</thead><tbody>
<tr><td>John</td><td>Muir</td><td>Yosemite</td><td></td></tr>
<tr><td colspan="2">Michael Tilson Thomas</td><td>San Francisco
Symphony</td><td>4158885555</td></tr>
<tr><td>Diane </td><td>Nicolini</td><td rowspan="2">KDFC</td><td
rowspan="2">415888KDFC</td></tr>
<tr><td>Bill</td><td>Lueth</td></tr>
</tbody>
</table>
```

, the "uebertag"

Finally, there is the `<div>`, the tag of all tags. When you run into a problem trying to fit things on the page where you want them, you will most likely solve it by inserting a number of `<div>` elements. Think of a `<div>` as a rectangular section of your page. You can even organize your page as a grid. The framework we will be using in the second half of the book is exactly that. It uses a 12-column grid.

Look at this very simple, yet not uncommon example:

```
<body>
<div id="header"></div>
<div id="container">
<div id="left"></div><div id="middle"></div><div id="right"></div>
</div>
<div id="footer"></div>
</body>
```

Just make this the body of a new HTML page, `minigrid.html`, and look at it. You will see ... nothing, because none of the `<div>` elements have any content, in which case they do not have any size. `<div>` elements are so-called block elements. We will cover this in great detail in the next chapter. Before we do that, we are going to conclude this chapter on HTML with a very important topic: **HTML entities**.

HTML entities

As we know, all tags begin with a `<` sign and end with a `>` sign. Just imagine you want to use one of those as part of your content. This just might confuse the browser. That is why we have HTML entities.

HTML entities are strings that begin with an ampersand and end with a semicolon.

- This represents the ampersand itself:

  ```
  &amp;
  ```

- A very useful HTML entity is the non-breaking space:

  ```
   
  ```

  It allows you to insert one or more spaces in you content. To use the `<` or `>` sign in your content, we have: `<` and `>`

- Also very useful are `&eur;`, for the Euro symbol, `&copy;` for the copyright sign, and `&reg;` for the Registered Trademark sign.
- Non-English characters can be represented as HTML entities as well, for example, `&eacute;` for é, `&egrave;` for è, and `&ecirc;` for ê.

We recommend you look up some of the online references if you want to see a complete list.

HTML5-specific tags

HTML5 introduced a number of new tags that can help you to add structure to your document, as they all have the names of common components of a document or site, such as header, footer, or article.

If you have been doing web development for a while, you will have used these names in all likelihood, but as a class to categorize `<div>` elements. So if you used `<div class="header">`, you can now use `<header>`. Or, to turn things around, if you already use `<header>`, your fallback plan to support non-HTML5-capable browsers could contain `<div class="header">`.

Here is a brief overview of what they are and how they can be used:

- `<header>`: This is used to contain the headline for a page or section. It typically contains a company logo and navigational elements.
- `<footer>`: Footers typically contain links to other related information, contact info, and copyright statements. Make sure you keep the latter up-to-date. People will not trust the information on a site that has a date of two years ago.
- `<nav>`: This container can be used for the main navigation portion of your site.
- `<aside>`: This tag is very useful to place the component of your side that often is placed on the left, next to everything else.
- `<article>` and `<section>`: These two are useful to better organize your document. You can use them for blog posts or, as the names suggest, articles or sections.

Summary

In this chapter, you learned the basics of HTML, the first of several languages we will use to do web development. HTML uses tags such as `<div>` and tags can also have attributes, for example, `<p class="blue">blue paragraph</p>`. All these tags combined on a page, an HTML page, form the building blocks of a website. Rather than giving an exhaustive list and description of all available HTML elements of tags, we walked you through the most common and useful ones.

HTML elements are used to add content to a website, not to give the site a certain look. Colors, the background and foreground, letter types, borders around images, and many more visible features of a site are handled through style sheets and another language. That language is CSS and that is the topic of our next chapter.