Introduction to HTML

Qitian Liao

University of California, Berkeley

Contents

1 Introduction to HTML **1**

1.1 Introduction to HTML . . . 1
1.2 HTML Anatomy . . . 1
1.3 The Body . . . 2
1.4 HTML Structure . . . 2
1.5 Headings . . . 3
1.6 Divs . . . 3
1.7 Attributes . . . 4
1.8 Displaying Text . . . 4
1.9 Styling Text . . . 4
1.10 Line Breaks . . . 5
1.11 Unordered Lists . . . 5
1.12 Ordered Lists . . . 6
1.13 Images . . . 6
1.14 Image Alts . . . 7
1.15 Videos . . . 7
1.16 Review . . . 7

2 HTML Document Standards **9**

2.1 Preparing for HTML . . . 9
2.2 The <html> tag . . . 9
2.3 The Head . . . 9
2.4 Page Titles . . . 10
2.5 Where Does the Title Appear? . . . 10
2.6 Linking to Other Web Pages . . . 11
2.7 Opening Links in a New Window . . . 11
2.8 Linking to Relative Page . . . 11
2.9 Linking At Will . . . 12
2.10 Linking to Same Page . . . 13
2.11 Whitespace . . . 13

1 Introduction to HTML

Now we will learn the basic structure of an HTML document.

1.1 Introduction to HTML

HTML is the skeleton of all web pages. It is often the first language learned by developers, marketers, and designers and is core to front-end development work. HTML provides structure to the content appearing on a website, such as images, text, or videos. Right-click on any page on the internet, choose "Inspect," and we will see HTML in a panel of your screen.

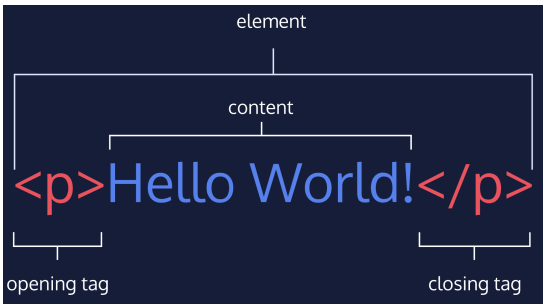
HTML stands for HyperText Markup Language:

- A *markup* language is a computer language that defines the structure and presentation of raw text.
- In HTML, the computer can interpret *raw text* that is wrapped in HTML elements.
- *HyperText* is text displayed on a computer or device that provides access to other text through links, also known as *hyperlinks*.

Learning HTML is the first step in creating websites, but even a bit of knowledge can help us inject code snippets into newsletter, blog or website templates. As we continue learning, we can layer HTML with CSS and JavaScript to create visually compelling and dynamic websites. Now, we are going to focus on how to add and modify basic content on a page, like text, images, and videos.

1.2 HTML Anatomy

HTML is composed of *elements*. These elements structure the webpage and define its content. Let us take a look at how they are written.



The diagram above displays an HTML paragraph element. As we can see, the paragraph element is made up of:

- An *opening tag* (`<p>`)
- The content ("Hello World!" text)
- A *closing tag* (`</p>`)

A *tag* and the *content* between it is called an HTML element. There are many tags that we can use to organize and display text and other types of content, like images.

Let us quickly review each part of the element pictured:

- HTML element (or simply, element) — a unit of content in an HTML document formed by HTML tags and the text or media it contains.
- HTML Tag — the element name, surrounded by an opening (<) and closing (>) angle bracket.
- Opening Tag — the first HTML tag used to start an HTML element. The tag type is surrounded by opening and closing angle brackets.
- Content — The information (text or other elements) contained between the opening and closing tags of an HTML element.
- Closing tag — the second HTML tag used to end an HTML element. Closing tags have a forward slash (/) inside of them, directly after the left angle bracket.

1.3 The Body

One of the key HTML elements we use to build a webpage is the *body* element. Only content inside the opening and closing body tags can be displayed to the screen. Here is what opening and closing body tags look like:

```
<body>

</body>
```

Once the file has a body, many different types of content – including text, images, and buttons – can be added to the body.

```
<body>
  <p>What's up, doc?</p>
</body>
```

1.4 HTML Structure

HTML is organized as a collection of family tree relationships. As you saw in the last exercise, we placed <p> tags within <body> tags. When an element is contained inside another element, it is considered the *child* of that element. The child element is said to be *nested* inside of the *parent* element.

```
<body>
  <p>This paragraph is a child of the body</p>
</body>
```

In the example above, the <p> element is nested inside the <body> element. The <p> element is considered a child of the <body> element, and the <body> element is considered the parent. Notice that we have added two spaces of indentation (using the space bar) for better readability.

Since there can be multiple levels of nesting, this analogy can be extended to grandchildren, great-grandchildren, and beyond. The relationship between elements and their ancestor and descendent elements is known as *hierarchy*.

Let us consider a more complicated example that uses some new tags:

```
<body>
  <div>
    <h1>Sibling to p, but also grandchild of body</h1>
```

```
    <p>Sibling to h1, but also grandchild of body</p>
  </div>
</body>
```

In this example, the `<body>` element is the parent of the `<div>` element. Both the `<h1>` and `<p>` elements are children of the `<div>` element. Because the `<h1>` and `<p>` elements are at the same level, they are considered siblings and are both grandchildren of the `<body>` element.

Understanding HTML hierarchy is important because child elements can inherit behavior and styling from their parent element. We will learn more about webpage hierarchy when you start digging into CSS.

1.5 Headings

Headings in HTML are similar to headings in other types of media. For example, in newspapers, large headings are typically used to capture a reader's attention. Other times, headings are used to describe content, like the title of a movie or an educational article.

HTML follows a similar pattern. In HTML, there are six different *headings*, or *heading elements*. Headings can be used for a variety of purposes, like titling sections, articles, or other forms of content.

The following is the list of heading elements available in HTML. They are ordered from largest to smallest in size.

1. `<h1>` — used for main headings. All other smaller headings are used for subheadings.
2. `<h2>`
3. `<h3>`
4. `<h4>`
5. `<h5>`
6. `<h6>`

The following example code uses a headline intended to capture a reader's attention. It uses the largest heading available, the main heading element:

```
<h1>BREAKING NEWS</h1>
```

1.6 Divs

One of the most popular elements in HTML is the `<div>` element. `<div>` is short for "division" or a container that divides the page into sections. These sections are very useful for grouping elements in your HTML together.

```
<body>
  <div>
    <h1>Why use divs?</h1>
    <p>Great for grouping elements!</p>
  </div>
</body>
```

`<div>`s can contain any text or other HTML elements, such as links, images, or videos. Remember to always add two spaces of indentation when you nest elements inside of `<div>`s for better readability.

1.7 Attributes

If we want to expand an element's tag, we can do so using an *attribute*. Attributes are content added to the opening tag of an element and can be used in several different ways, from providing information to changing styling. Attributes are made up of the following two parts:

- The *name* of the attribute
- The *value* of the attribute

One commonly used attribute is the `id`. We can use the `id` attribute to specify different content (such as `<div>`s) and is really helpful when you use an element more than once. `id`s have several different purposes in HTML, but for now, we will focus on how they can help us identify content on our page.

When we add an `id` to a `<div>`, we place it in the opening tag:

```
<div id="intro">
  <h1>Introduction</h1>
</div>
```

1.8 Displaying Text

If you want to display text in HTML, you can use a *paragraph* or *span*:

- *Paragraphs* (`<p>`) contain a block of plain text.
- `<span>` contains short pieces of text or other HTML. They are used to separate small pieces of content that are on the same line as other content.

Take a look at each of these elements in action below:

```
<div>
  <h1>Technology</h1>
</div>
<div>
  <p><span>Self-driving cars</span> are anticipated to
  replace up to 2 million jobs over the next two decades.</p>
</div>
```

In the example above, there are two different `<div>`. The second `<div>` contains a `<p>` with `<span>Self-driving cars</span>`. This `<span>` element separates "Self-driving cars" from the rest of the text in the paragraph.

It is best to use a `<span>` element when you want to target a specific piece of content that is inline, or on the same line as other text. If you want to divide your content into blocks, it is better to use a `<div>`.

1.9 Styling Text

You can also style text using HTML tags. The `<em>` tag emphasizes text, while the `<strong>` tag highlights important text.

Later, when we begin to style websites, we will decide how you want browsers to display content within `<em>` and `<strong>` tags. Browsers, however, have built-in style sheets that will generally style these tags in the following ways:

- The tag will generally render as italic emphasis.
- The will generally render as bold emphasis.

Take a look at each style in action:

```
<p><strong>The Nile River</strong> is the <em>longest</em>
river in the world, measuring over 6,850 kilometers long
(approximately 4,260 miles).</p>
```

In this example, the and tags are used to emphasize the text to produce the following:

The Nile River is the *longest* river in the world, measuring over 6,850 kilometers long (approximately 4,260 miles).

As we can see, "The Nile River" is bolded and "longest" is in italics.

1.10 Line Breaks

The spacing between code in an HTML file does not affect the positioning of elements in the browser. If you are interested in modifying the spacing in the browser, you can use HTML's *line break* element:
.

The line break element is unique because it is only composed of a starting tag. You can use it anywhere within your HTML code and a line break will be shown in the browser.

```
<p>The Nile River is the longest river <br> in the world,
measuring over 6,850 <br> kilometers long (approximately
4,260 <br> miles).</p>
```

The code in the example above will result in an output that looks like the following:

The Nile River is the longest river
in the world, measuring over 6,850
kilometers long (approximately 4,260
miles).

1.11 Unordered Lists

In addition to organizing text in paragraph form, you can also display content in an easy-to-read list.

In HTML, you can use an *unordered list* tag (<ul>) to create a list of items in no particular order. An unordered list outlines individual list items with a bullet point.

The <ul> element should not hold raw text and will not automatically format raw text into an unordered list of items. Individual list items must be added to the unordered list using the <li> tag. The <li> or list item tag is used to describe an item in a list.

```
<ul>
  <li>Limes</li>
  <li>Tortillas</li>
  <li>Chicken</li>
</ul>
```

In the example above, the list was created using the `<ul>` tag and all individual list items were added using `<li>` tags.

The output will look like this:

- Limes
- Tortillas
- Chicken

1.12 Ordered Lists

Ordered lists (`<ol>`) are like unordered lists, except that each list item is numbered. They are useful when you need to list different steps in a process or rank items for first to last.

You can create the ordered list with the `<ol>` tag and then add individual list items to the list using `<li>` tags.

```
<ol>
  <li>Preheat the oven to 350 degrees.</li>
  <li>Mix whole wheat flour, baking soda, and salt.</li>
  <li>Cream the butter, sugar in separate bowl.</li>
  <li>Add eggs and vanilla extract to bowl.</li>
</ol>
```

The output will look like this:

---

1. Preheat the oven to 350 degrees.
2. Mix whole wheat flour, baking soda, and salt.
3. Cream the butter, sugar in separate bowl.
4. Add eggs and vanilla extract to bowl.

1.13 Images

All of the elements we have learned about so far (headings, paragraphs, lists, and spans) share one thing in common: they are composed entirely of text. What if we want to add content to our web page that is not composed of text, like images?

The `<img>` tag allows us to add an image to a web page. Most elements require both opening and closing tags, but the `<img>` tag is a self-closing tag. Note that the end of the `<img>` tag has a forward slash `/`. Self-closing tags may include or omit the final slash — both will render properly.

```
<img src="image-location.jpg" />
```

The `<img>` tag has a required *attribute* called `src`. The `src` attribute must be set to the image's *source*, or the location of the image. In this case, the value of `src` must be the *uniform resource locator* (URL) of the image. A URL is the web address or local address where a file is stored.

1.14 Image Alts

Part of being an exceptional web developer is making our site accessible to users of all backgrounds. In order to make the Web more inclusive, we need to consider what happens when assistive technologies such as screen readers come across image tags.

The `alt` attribute, which means alternative text, brings meaning to the images on our sites. The `alt` attribute can be added to the image tag just like the `src` attribute. The value of `alt` should be a description of the image.

```
<img src="#" alt="A field of yellow sunflowers" />
```

The `alt` attribute also serves the following purposes:

- If an image fails to load on a web page, a user can mouse over the area originally intended for the image and read a brief description of the image. This is made possible by the description you provide in the `alt` attribute.
- Visually impaired users often browse the web with the aid of screen reading software. When you include the alt attribute, the screen reading software can read the image's description out loud to the visually impaired user.
- The alt attribute also plays a role in Search Engine Optimization (SEO), because search engines cannot "see" the images on websites as they crawl the internet. Having descriptive alt attributes can improve the ranking of your site.

If the image on the web page is not one that conveys any meaningful information to a user (visually impaired or otherwise), the `alt` attribute should be left empty.

1.15 Videos

In addition to images, HTML also supports displaying videos. Like the `<img>` tag, the `<video>` tag requires a src attribute with a link to the video source. Unlike the `<img>` tag however, the `<video>` element requires an opening and a closing tag.

```
<video src="myVideo.mp4" width="320" height="240" controls>
  Video not supported
</video>
```

In this example, the video source (`src`) is `myVideo.mp4`. The source can be a video file that is hosted alongside your webpage, or a URL that points to a video file hosted on another webpage.

After the `src` attribute, the `width` and `height` attributes are used to set the size of the video displayed in the browser. The `controls` attribute instructs the browser to include basic video controls: pause, play and skip.

The text, "Video not supported", between the opening and closing video tags will only be displayed if the browser is unable to load the video.

1.16 Review

Let us review what we have learned in this chapter:

- **HTML** stands for **H**yper**T**ext **M**arkup **L**anguage and is used to create the structure and content of a webpage.
- Most HTML elements contain opening and closing tags with raw text or other HTML tags between them.
- HTML elements can be nested inside other elements. The enclosed element is the child of the enclosing parent element.
- Any visible content should be placed within the opening and closing `<body>` tags.
- Headings and sub-headings, `<h1>` to `<h6>` tags, are used to enlarge text.
- `<p>`, `<span>` and `<div>` tags specify text or blocks.
- The `<em>` and `<strong>` tags are used to emphasize text.
- Line breaks are created with the `<br>` tag.
- Ordered lists (`<ol>`) are numbered and unordered lists (`<ul>`) are bulleted.
- Images (`<img>`) and videos (`<video>`) can be added by linking to an existing source.

Next, we will take the content that we have added to the website and transform it into an HTML document that is ready to go on the web.

2 HTML Document Standards

Now we will learn how to set up a proper HTML Document, how to link to other pages, and how to format our code for readability.

2.1 Preparing for HTML

Now it is time to learn how to set up an HTML file. HTML files require certain elements to set up the document properly. We can let web browsers know that we are using HTML by starting our document with a *document type declaration*.

The declaration looks like this:

```
<!DOCTYPE html>
```

This declaration is an instruction, and it must be the first line of code in your HTML document. It tells the browser what type of document to expect, along with what version of HTML is being used in the document. For now, the browser will correctly assume that the `html` in `<!DOCTYPE html>` is referring to HTML5, as it is the current standard.

In the future, however, a new standard will override HTML5. To make sure your document is forever interpreted correctly, always include `<!DOCTYPE html>` at the very beginning of your HTML documents.

Lastly, HTML code is always saved in a file with an **.html** extension.

2.2 The <html> tag

The `<!DOCTYPE html>` declaration provides the browser with two pieces of information (the type of document and the HTML version to expect), but it does not actually add any HTML structure or content.

To create HTML structure and content, we must add opening and closing `<html>` tags after declaring `<!DOCTYPE html>`:

```
<!DOCTYPE html>
<html>

</html>
```

Anything between the opening `<html>` and closing `</html>` tags will be interpreted as HTML code. Without these tags, it is possible that browsers could incorrectly interpret your HTML code.

2.3 The Head

So far we have done two things to set up the file properly:

- Declared to the browser that your code is HTML with `<!DOCTYPE html>`
- Added the HTML element (`<html>`) that will contain the rest of your code.

Now, let us also give the browser some information about the page itself. We can do this by adding a `<head>` element.

The `<head>` element is part of the HTML metaphor as the `<body>` tag. It goes above the `<body>` element.

The <head> element contains the *metadata* for a web page. Metadata is information about the page that is not displayed directly on the web page. Unlike the information inside of the <body> tag, the metadata in the head is information about the page itself.

The opening and closing head tags typically appear as the first item after our first HTML tag:

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

2.4 Page Titles

Let us investigate what kind of metadata about the web page can the <head> element contain.

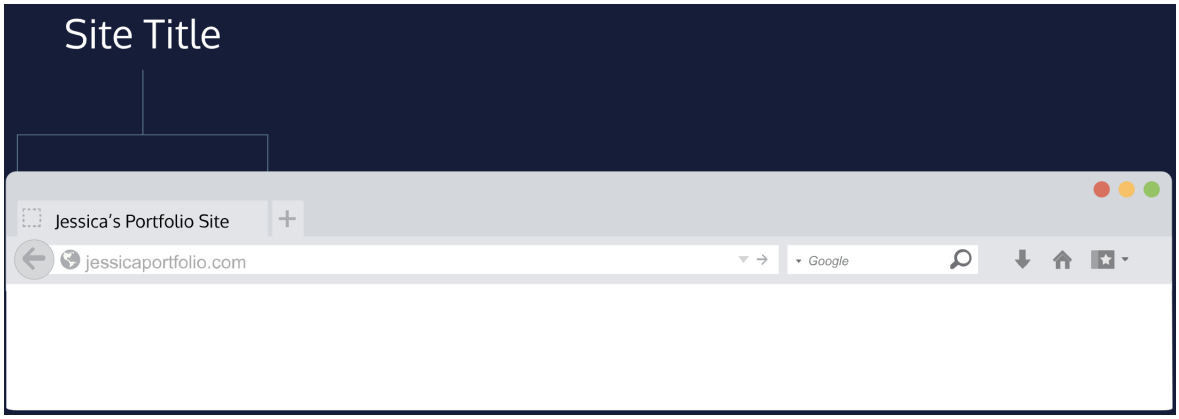
A browser's tab displays the *title* specified in the <title> tag. The <title> tag is always inside of the <head>.

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Coding Journal</title>
  </head>
</html>
```

If we were to open a file containing the HTML code in the example above, the browser would display the words My Coding Journal in the title bar (or in the tab's title).

2.5 Where Does the Title Appear?

Our title would appear as depicted in the diagram below.



So far, we have learned about:

- <!DOCTYPE html>, the declaration specifying the version of HTML for the browser
- The <html> tags that enclose all of your HTML code
- The <head> tag that contains the metadata of a webpage, such as its <title>

Next, we will learn about new types of elements that go inside the body.

2.6 Linking to Other Web Pages

One of the powerful aspects of HTML (and the Internet), is the ability to *link* to other web pages.

We can add links to a web page by adding an *anchor* element `<a>` and including the text of the link in between the opening and closing tags.

```
<a>This Is A Link To Wikipedia</a>
```

Technically, the link in the example above is incomplete. The link above cannot work because there is no URL that will lead users to the actual Wikipedia page.

The anchor element in the example above is incomplete without the `href` attribute. This attribute stands for *hyperlink reference* and is used to link to a *path*, or the address to where a file is located (whether it is on your computer or another location). The paths provided to the `href` attribute are often URLs.

```
<a href="https://www.wikipedia.org/">This Is A Link To Wikipedia</a>
```

In the example above, the `href` attribute has been set to the value of the URL `https://www.wikipedia.org/`. The example now shows the correct use of an anchor element.

When reading technical documentation, we may come across the term *hyperlink*. This is simply the technical term for link, and these terms are often used interchangeably.

2.7 Opening Links in a New Window

Have you ever clicked on a link and observed the resulting web page open in a new browser window? If so, you can thank the `<a>` element's `target` attribute.

The `target` attribute specifies how a link should open.

It is possible that one or more links on your web page link to an entirely different website. In that case, you may want users to read the linked website, but hope that they return to your web page. This is exactly when the `target` attribute is useful.

For a link to open in a new window, the `target` attribute requires a value of `_blank`. The `target` attribute can be added directly to the opening tag of the anchor element, just like the `href` attribute.

```
<a href="https://en.wikipedia.org/wiki/Brown_bear"
target="_blank">The Brown Bear</a>
```

In the example above, setting the `target` attribute to "`_blank`" instructs the browser to open the relevant Wikipedia page in a new window.

In this exercise, we have used the terminology "open in a new window." It is likely that you are using a modern browser that opens up websites in new *tabs*, rather than new windows. Before the advent of browsers with tabs, additional browser windows had to be opened to view more websites. The `target="_blank"` attribute, when used in modern browsers, will open new websites in a new tab.

2.8 Linking to Relative Page

Thus far we have learned how to link to external web pages. Many sites also link to internal web pages like Home, About, and Contact.

Before we learn how to link between internal pages, let us establish where our files are stored. When making multi-page static websites, web developers often store HTML files in the root directory, or a main folder where all the files for the project are stored. As the size of the projects you create grows, you may use additional folders within the main project folder to organize your code.

```
project-folder/
|__ about.html
|__ contact.html
|__ index.html
```

The example above shows three different files — **about.html**, **contact.html**, and **index.html** in one folder.

HTML files are often stored in the same folder, as shown in the example above. If the browser is currently displaying **index.html**, it also knows that **about.html** and **contact.html** are in the same folder. Because the files are stored in the same folder, we can link web pages together using a *relative path*.

```
<a href="./contact.html">Contact</a>
```

In this example, the `<a>` tag is used with a relative path to link from the current HTML file to the `contact.html` file in the same folder. On the web page, `Contact` will appear as a link.

A relative path is a filename that shows the path to a *local file* (a file on the same website, such as `./index.html`) versus an absolute path (a full URL, like `https://www.codecademy.com/learn/learn-html` which is stored in a different folder). The `./` in `./index.html` tells the browser to look for the file in the current folder.

2.9 Linking At Will

You have probably visited websites where not all links were made up of text. Maybe the links you clicked on were images or some other form of content.

So far, we have added links that were made up of only text, like the following:

```
<a href="https://en.wikipedia.org/wiki/Opuntia"
target="_blank">Prickly Pear</a>
```

Text-only links, however, would significantly decrease our flexibility as web developers.

Thankfully, HTML allows us to turn nearly any element into a link by wrapping that element with an anchor element. With this technique, it is possible to turn images into links by simply wrapping the `<img>` element with an `<a>` element.

```
<a href="https://en.wikipedia.org/wiki/Opuntia"
target="_blank"><img
src="https://www.Prickly_Pear_Closeup.jpg" alt="A red
prickly pear fruit"/></a>
```

In the example above, an image of a prickly pear has been turned into a link by wrapping the outside of the `<img>` element with an `<a>` element.

2.10 Linking to Same Page

At this point, we have all the content we want on our page. Since we have so much content, it does not all fit on the screen. How do we make it easier for a user to jump to different portions of our page?

When users visit our site, we want them to be able to click a link and have the page automatically scroll to a specific section.

In order to link to a target on the same page, we must give the target an *id*, like this:

```
<p id="top">This is the top of the page!</p>
<h1 id="bottom">This is the bottom! </h1>
```

In this example, the `<p>` element is assigned an `id` of "top" and the `<h1>` element is assigned "bottom." An `id` can be added to most elements on a webpage.

An `id` should be descriptive to make it easier to remember the purpose of a link. The target link is a string containing the `#` character and the target element's `id`.

```
<ol>
  <li><a href="#top">Top</a></li>
  <li><a href="#bottom">Bottom</a></li>
</ol>
```

In the example above, the links to `<p id="top">` and `<h1 id="bottom">` are embedded in an ordered list. These links appear in the browser as a numbered list of links. An `id` is especially helpful for organizing content belonging to a `div`.

2.11 Whitespace

Now we will focus on some tools developers use to make code easier to interpret.

As the code in an HTML file grows, it becomes increasingly difficult to keep track of how elements are related. Programmers use two tools to visualize the relationship between elements: *whitespace* and *indentation*.

Both tools take advantage of the fact that the position of elements in a browser is independent of the amount of whitespace or indentation in an HTML file.

For example, if we wanted to increase the space between two paragraphs on your web page, you would not be able to accomplish this by adding space between the paragraph elements in the HTML file. The browser ignores whitespace in HTML files when it renders a web page, so it can be used as a tool to make code easier to read and follow.

What makes the example below difficult to read?

```
<body><p>Paragraph 1</p><p>Paragraph 2</p></body>
```

We have to read the entire line to know what elements are present. Compare the example above to this:

```
<body>
    <p>Paragraph 1</p>
    <p>Paragraph 2</p>
</body>
```

This example is easier to read, because each element is on its own line. While the first example required you to read the entire line of code to identify the elements, this example makes it easy to identify the body tag and two paragraphs.

A browser renders both examples the same way:

```
Paragraph 1
Paragraph 2
```

Next we will learn how to use indentation to help visualize nested elements.