1.7.4 Set Up the HTML File

Before we add the privacy policy content, we need to create the skeleton of the HTML document. This means we need to get our starting HTML tags in place.

Again, think back to Lesson 1 before any Run Buddy-specific content was added. That page had the following:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Run Buddy</title>
  </head>
  <body>
  </body>
</html>
```

NERD NOTE

Creating the skeleton of an HTML document is also known as **scaffolding**.

When we're done scaffolding `privacy-policy.html`, we'll need to make a couple of edits. Because we're creating a new page, we need to update the `<title>` to reflect that. The information that's currently between the `<title>` tags (i.e., Run Buddy) isn't incorrect, but it isn't as descriptive as it could be. We want to be explicit about exactly what page we're on.

A descriptive title lets the user know where they are on the site. It's also important for accessibility and search engine optimization.

Edit the `<title>` to be the following:

```
<title>Privacy Policy - Run Buddy</title>
```

HIDE PRO TIP

The `<title>` element's content is what appears in the browser's tab, so it's good practice to structure the content as `[page title] - [site title]`. Page titles should be descriptive but also concise because Google cuts off search result titles at around 60 characters.

To learn more, check out the **MDN web docs on creating SEO-friendly page titles** (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/title#Page_titles_and_SEO).

Take another look at the mock-up of the Privacy Policy page and compare it to the Run Buddy landing page you just finished. Do you see any pieces you can reuse? It can be confusing for visitors to experience different styles throughout a single website, so most sites have similar—if not identical—components on every page.

NERD NOTE

A set of HTML code that builds out a part of the overall page's UI is commonly referred to as a **component**. The idea behind a component is that it can be reused in multiple places throughout a site.

Let's identify the reusable components and then copy and paste them into our new document:

- **`<header>`**: The entire `<header>` element can be copied over to the new page because its design and layout is the same on both pages. The only change we need to make is to edit the navigation `<a>` elements' `href` values. We need to add `./index.html` in front of the `#what-we-do` value (for example). Here's how that looks:

```
<!-- Do this to all of the <a> elements in privacy-policy.html -->
<a href="./index.html#what-we-do">What We Do</a>
```

Before we move on, let's think about that new value for `href`. In `index.html`, we simply provided a value of `#what-we-do`. That told the browser that, when clicked, it should route the user to another location within the page that has an `id` attribute with the value set to `what-we-do`.

We still want this functionality, but now that `<a>` element is on a different page. So how do we get the user from `privacy-policy.html` to `index.html`, specifically to that section with a corresponding `id`? As you can see by the edit in the code above, we include both locations at once. The value `./index.html#what-we-do` can almost be read in two parts:

1. Go to the `index.html` file at the location `./`, which means "in the current directory."

2. Once there, navigate within it to the element with an `id` of `what-we-do`.

Now that you know how to make these new `href` values work, go ahead and make sure the other three navigation `<a>` elements follow the same pattern.

REWIND

The `<a>` element routing you to a different location is an example of the "hypertext" in HTML.

Let's move on to the next section. The other content from our homepage that can be copied over to our privacy policy is the following:

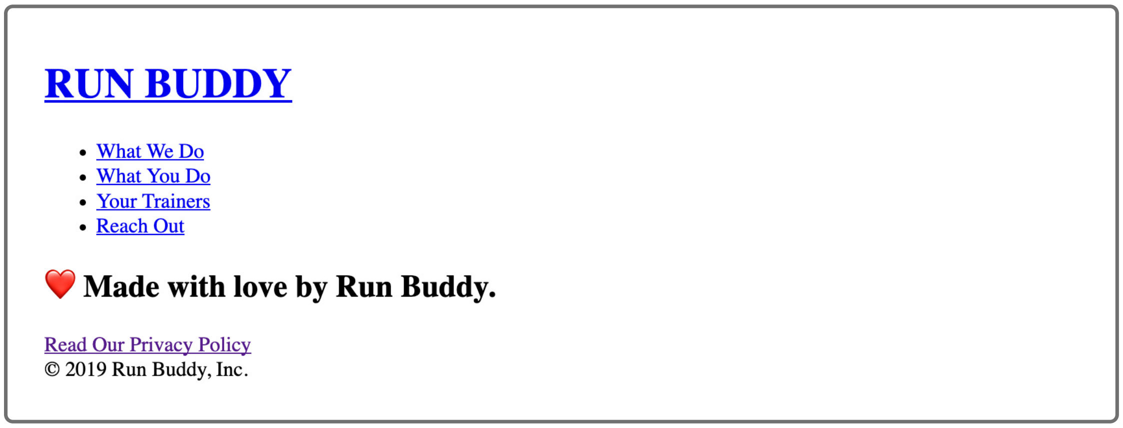
- **`<footer>`**: This can be copied directly into `privacy-policy.html`. Nothing needs to be edited here.
- **`<section class="hero">`**: We will reuse this, but we need to edit the content so it includes Run Buddy's privacy policy verbiage. For now, it is easier to go into `privacy-policy.html` and type the following right after the closing `</header>` tag:

```
<section class="hero">

</section>
```

We'll get to adding content to this section soon, but first let's save our work and open this file in the browser.

If your page doesn't look like the image below, review the previous steps to troubleshoot. The culprit is often something very simple but easy to overlook. Remember—even pros make mistakes!



RUN BUDDY

- What We Do
- What You Do
- Your Trainers
- Reach Out

❤️ Made with love by Run Buddy.

Read Our Privacy Policy
© 2019 Run Buddy, Inc.

Now that we have a second page in our website, how will the visitor get to it from the homepage? In `index.html`, we need to change the `<a>` tag's `href` value in the `<footer>` tag to have a value of `./privacy-policy.html`.

We can remove the `href` attribute from this element in `privacy-policy.html` entirely since we are already on this page and don't want users accidentally clicking the link and reloading the page for no reason.

HIDE PRO TIP

When you have an `<a>` element with an `href` value set to take the user to the same page they're already on, it has the potential to decrease the site's performance because it has to download and display all of the page's data again. Due to this, it is a good practice when possible to disable `<a>` elements that bring you to the page you're currently on when clicked.

It won't always be possible depending on the site's requirements, but it is something to keep an eye out for.

We've used relative pathing in our `<img>` and `<link>` tags to target other files in our project's folder structure, and now we're doing the same thing here to target the `privacy-policy.html` file that we created in the same directory as `index.html`. The biggest difference is how these HTML elements interact with the other file.

When we use `<img>` and `<link>` elements, we're telling another file to join this HTML file in some fashion. Those elements don't take us anywhere; they bring resources to us. When we use `<a>` elements, however, we are doing the opposite by saying "When I'm clicked, I'll leave this location and take you somewhere else in your browser." Again, all of these HTML elements exemplify the concept of **hypertext** in HTML.

DEEP DIVE

DEEP DIVE

We can use `<a>` elements to bring a user to a variety of resources and locations. They could bring them to a page in the site or another website entirely, which is the most popular use case, but they can also be used to open photos, PDFs, audio files, etc. Pretty much anything that can be opened in a browser can be used as an `href` value.

Test it for yourself sometime and set up an `<a>` that takes you to the path of a style sheet. When you click it, you'll be shown the contents of the CSS file you pointed it to.

Now let's take care of the content inside `<section class="hero">`. As you can see, we removed the sign-up form. We will repurpose this section to hold the page title by editing it to look like this:

```html
<section class="hero">
  <h2 class="page-title">
    Privacy Policy
  </h2>
</section>
```

Even though we aren't building any more HTML pages for this project, this section has now been repurposed so that if we were, we could reuse it!

ON THE JOB

It is not uncommon for a web-based project to be considered complete, only for a boss or client to come back and ask that more be added to it. These additions can come in the form of simple edits/additions to the existing project files—which is typically an easy change—or they could involve creating more pages, features, or even functionality.

That's why, when building new projects, it is a good practice to set up general styles and HTML content layouts that can easily be reused across new sections rather than having to start each new piece from scratch.