1.3.3 Build the Hero Section's Structure

The section we're about to build is sometimes considered the most important one in any website because it will be seen by every visitor, even those who visit for only a few seconds or who arrive by accident.

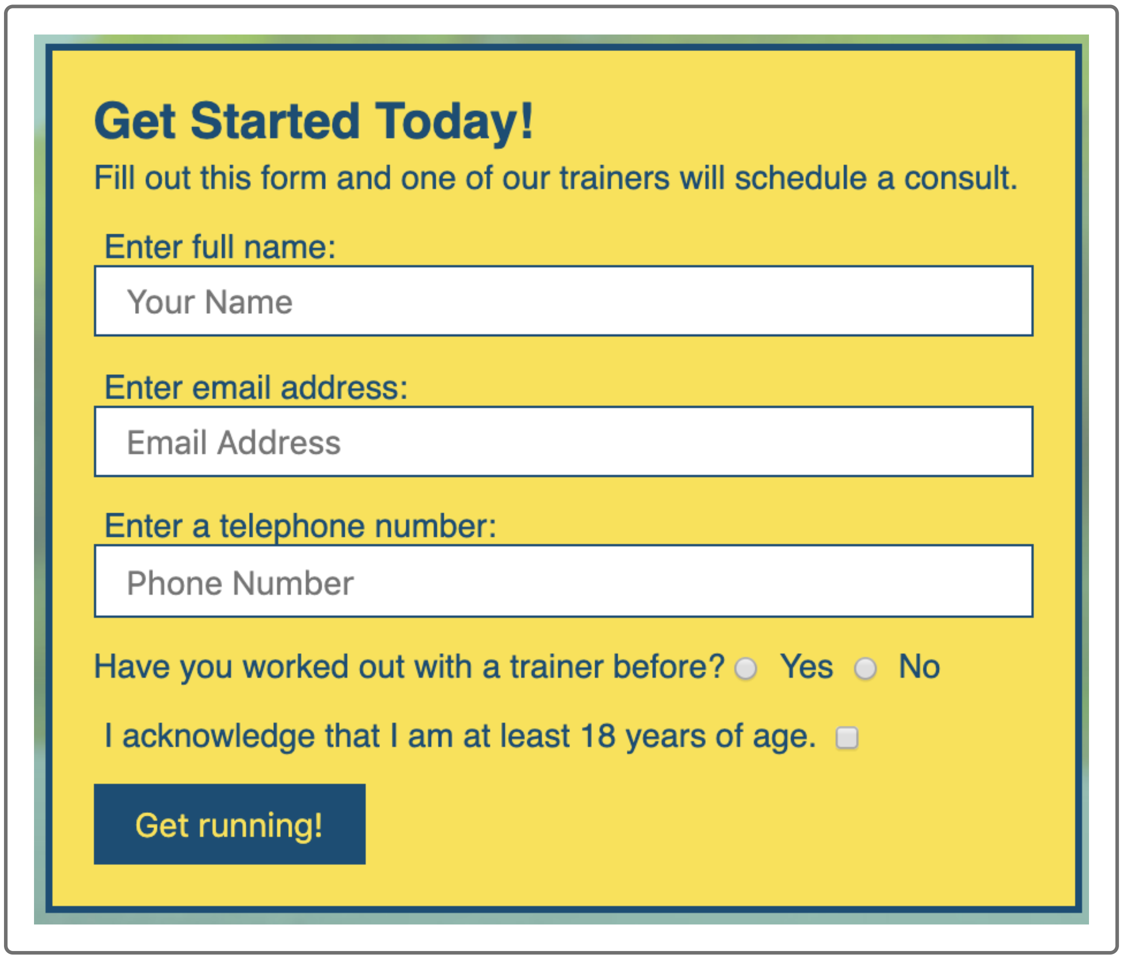
This section contains something called a **hero image**. In web terms, this refers to a large banner image placed prominently at the top of the page. The placement has a specific task: to pique the visitor's curiosity enough to get them reading, scrolling, clicking, and engaging with the website. Think of the hero section as the banner, headline, or billboard for a website.

As usual, before we can add the styling pizzazz, we need to create the HTML that contains the structure and content.

Let's build on the `<section>` element right below the closing `</header>` tag that we created in Lesson 1. This will act as the parent element that contains the children elements, the heading, and sign-up form. Let's start with the form.

Add the Sign-Up Form Container

Let's take a look at the sign-up form the client wants to see in this image:



Get Started Today!

Fill out this form and one of our trainers will schedule a consult.

Enter full name:

Your Name

Enter email address:

Email Address

Enter a telephone number:

Phone Number

Have you worked out with a trainer before? ○ Yes ○ No

I acknowledge that I am at least 18 years of age. ☐

PAUSE

From the mock-up, can you identify some HTML elements that are needed to create the sign-up form?

Initially, we'll need a box (`<div>`) that contains a heading (`<h2>`), paragraph text (`<p>`), input fields (`<input>`), and a button (`<button>`).

Hide Answer

The first thing we need to do is add a `<div>` element nested entirely inside the `<section>` element.

One important HTML rule is that both the opening and closing tags of an element must be completely contained inside the parent element.

This is the wrong way to nest elements:

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

Notice that only an opening tag for the `<div>` element is inside the `<section>` element; the closing tag should be there too.

In `index.html`, add the `<div>` element for the form right after the closing `</header>` tag. It's good practice to indent the child element. That section should look like this code when you're done:

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

IMPORTANT

> The browser doesn't care if lines are indented or if every element is on the same line. Nonetheless, indentation is an organizational practice that helps you and other developers read and understand the code, and is a great habit to get into.

Now that we have a `<div>` element, let's insert its child elements, including a heading for this section and some text.

Add elements and content for the heading and paragraph text inside the `<div>` element:

```
<section>
  <div>
    <h3>Get Started Today</h3>
    <p>Fill out this form and one of our trainers will schedule a cons
  </div>
</section>
```

You'll notice that we chose an `<h3>` element for the heading. But why not an `<h1>` or `<h4>`? Heading numbers represent the scope and significance of the heading. The browser makes an `<h1>` element the largest and boldest heading on the page, followed by the `<h2>`, then the `<h3>`, and so on.

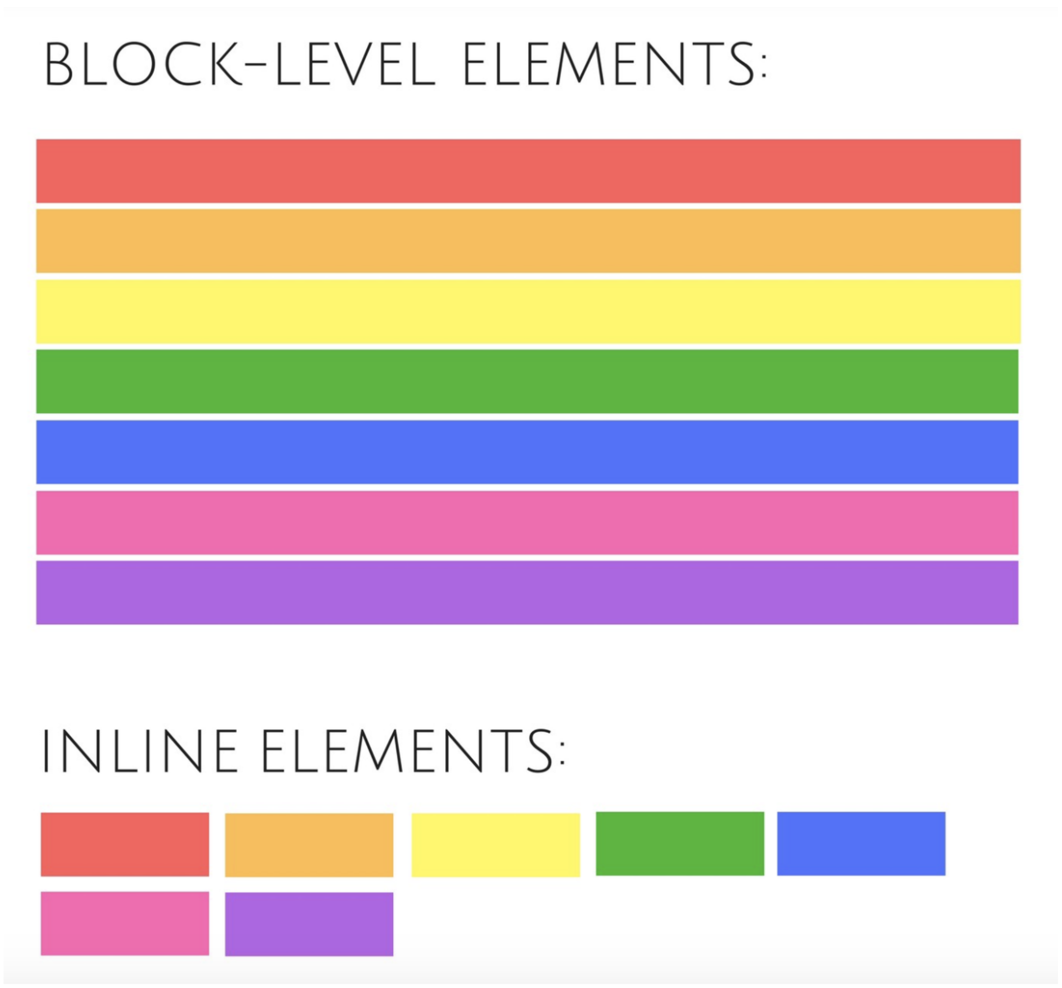
The main heading of our page is an `<h1>`, and section headers are `<h2>`. Because this heading is less significant than an `<h2>`, `<h3>` is the right choice.

IMPORTANT

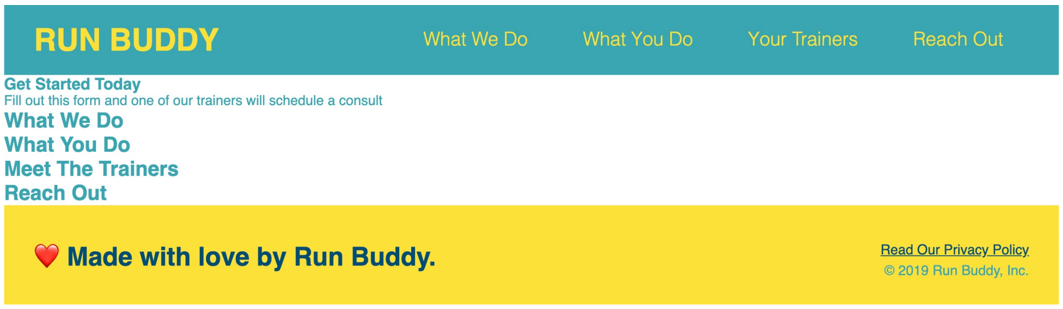
> To aid accessibility, headings should fall in sequential order; this helps screen readers interpret the relative importance of content on the page.

The `<p>` element means "paragraph" and is used to render the text in the page's body. By default, the browser declares a `<p>` to be a block-level element, but—unlike a heading element— won't assign any changes to font size or weight.

Here is an image to help show the difference between a block-level and inline-level display:



Your page should now look something like this image:



Get Started Today
Fill out this form and one of our trainers will schedule a consult
What We Do
What You Do
Meet The Trainers
Reach Out

❤️ Made with love by Run Buddy.
Read Our Privacy Policy
© 2019 Run Buddy, Inc.

This isn't quite what we're looking for stylistically, but the structure and content is set up the way we need it. Just like when building a house where the foundation and frame must go up before the rest can be built, the decorating (aka CSS styling!) always comes last.

Create the Sign-Up Form

It's time to add the sign-up form. Forms are a major point of interaction between users and websites. They invite and enable users to send data to the website, typically directly to the server.

The importance of our sign-up form is emphasized by its position at the top of the page. The form's ultimate goal is to generate sales leads and convert visitors into customers. Are you ready to learn how to create it? Let's do it!

First let's create the sign-up form container by using the `<form>` element. Within this `<form>` will be child elements such as `<input>` and `<label>`. `<input>` elements come in many flavors, such as open-ended (a text field that lets the user enter freeform text) and closed-ended (a radio button or checkbox for the user to select or check).

Let's add the form's HTML code to our hero's `<section>` element so it looks like this:

```
<section>
  <div>
    <h3>Get Started Today</h3>
    <p>Fill out this form and one of our trainers will schedule a cons
  <!-- Add form element -->
  <form></form>
  </div>
</section>
```

ON THE JOB

A sign-up form is a **call to action**, or **CTA**. The main purpose of a CTA is to encourage users via a story, advertisement, or dazzling piece of content to do something. CTAs play a vital role in converting a visitor into a sales lead—that's why we want it at the top of the page!

Add Text Input Fields

Let's add the text fields to the form. These are where the user will enter their name and contact information.

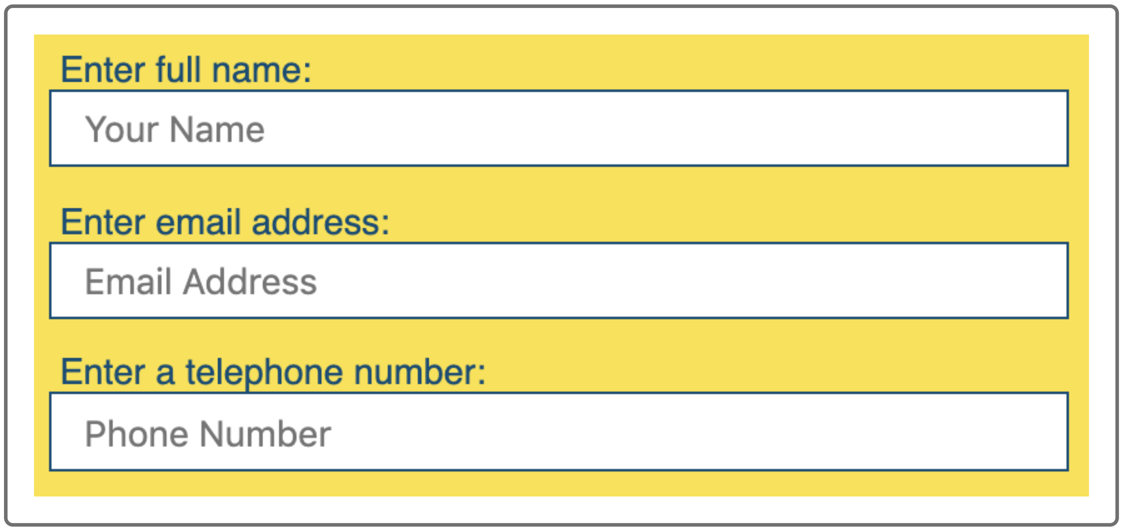
PAUSE

How many text fields are needed in the sign-up form?

We need three text fields: name, email address, and phone number.

Hide Answer

Let's take a look at these fields in the following image:



We'll start by adding the following HTML to the `<form>` element. Look closely at how we pair the `<input>` and corresponding `<label>` elements and their attributes:

```
<form>
 <label for="name">Enter full name:</label>
 <input type="text" placeholder="Your Name" name="name" id="name" />
</form>
```

The `<label>` text not only offers a visual directive of what to enter but also programmatically links to the associated `<input>`. When a user with a visual impairment uses a screen reader, the label will be read out when they focus on the input field. Labels also make it easier to fill out forms on mobile devices; clicking on the label will target the focus to the associated input field, which can be hard to do manually on a small screen.

Let's break down the attributes in the `<input>` element:

- **`for`** is an attribute in the `<label>` that programmatically links to the `id` attribute in the `<input>` element.
- **`type`** relates to what kind of input element we are using. Here we want a text field, which is also the default value.
- **`placeholder`** offers a hint or label within the text field, but will not be submitted if this field if left blank.
- **`name`** identifies the element so the response can be referenced after the form is submitted.

Given the first text input field for name, can you code the fields for email and phone number? Remember that the label and inputs will need the attributes for `name`, `for`, `type`, `id`, and `placeholder`.

When you're done, the HTML for these should look like this:

```
 <label for="email">Enter email address:</label>
 <input type="text" placeholder="Email Address" name="email" id="email
 <label for="phone">Enter a telephone number:</label>
 <input type="text" placeholder="Phone Number" name="phone" id="phone"
```

PAUSE

At the end of the `<input />` tag is a forward slash. What is this slash for?

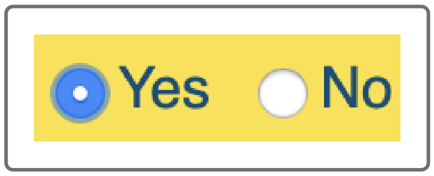
This is known as a self-closing tag and is a common sight in HTML. Some elements, like `<input>`, do not contain any content or child elements, so the closing tag immediately follows the opening tag. Thus this shortcut was created to self-close the opening tag.

Hide Answer

Add Radio Button Input

Next, we will create inputs for a radio button and checkbox. A **radio button** allows a single selection to be made from a list of choices and is typically a small circle that gets filled when clicked.

Here's an image of what this looks like:



PAUSE

Use your powers of Google to find some information about the function and syntax of a radio button.

Google will surely offer many search results, but the **MDN web docs article on radio buttons (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/radio)** should be a favorite because MDN is more likely to update when technologies change. Keep in mind that the best online documents to learn from should come from official sources like this one.

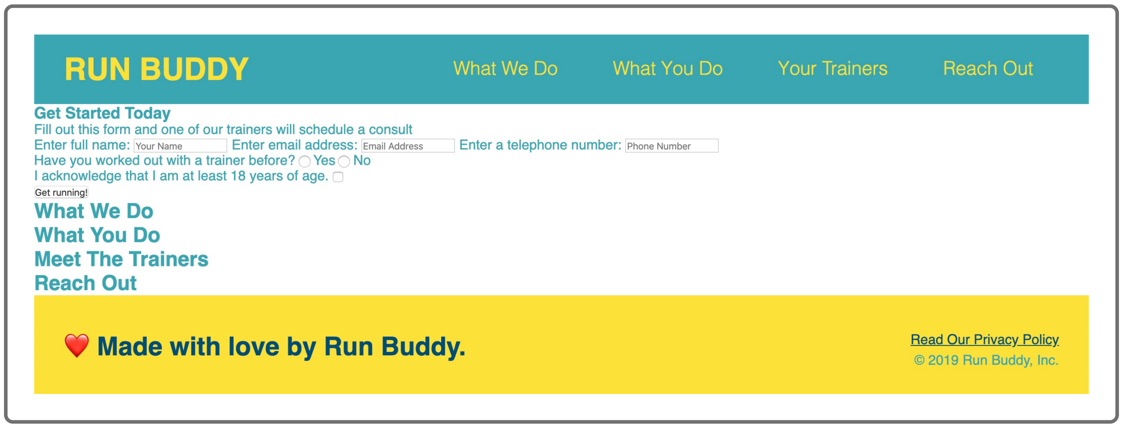
Hide Answer

So a `radio` type input element, commonly referred to as a radio button, is typically used in groups using the `name` attribute as the group reference, and each radio button must have the same `name` value to be considered in the same group.

Let's type some code and then explain the attributes afterwards. Add the `<p>` element below the text input fields you just created (the entire form's HTML is shown):

```
<form>
  <label for="name">Enter full name:</label>
  <input type="text" placeholder="Your Name" name="name" id="name" />
  <label for="email">Enter email address:</label>
  <input type="text" placeholder="Email Address" name="email" id="emai
  <label for="phone">Enter a telephone number:</label>
  <input type="text" placeholder="Phone Number" name="phone" id="phone
  <p>
    Have you worked out with a trainer before?
    <input type="radio" name="trainer-confirm" id="trainer-yes" />
    <label for="trainer-yes">Yes</label>
    <input type="radio" name="trainer-confirm" id="trainer-no" />
    <label for="trainer-no">No</label>
  </p>
</form>
```

Save the file and then see how it looks in the browser. The page should look like this image now:


RUN BUDDY What We Do What You Do Your Trainers Reach Out

Get Started Today
Fill out this form and one of our trainers will schedule a consult
Enter full name: Your Name Enter email address: Email Address Enter a telephone number: Phone Number
Have you worked out with a trainer before? Yes No
I acknowledge that I am at least 18 years of age.
Get running!
What We Do
What You Do
Meet The Trainers
Reach Out

Made with love by Run Buddy.
Read Our Privacy Policy
© 2019 Run Buddy, Inc.

Add Radio Button Layout

We've wrapped both of the answers, including both radio buttons, in a `<p>` element that also includes the question. Because the radio buttons are inline elements, the answer selections will render from left to right on the same line as our question, just like it does in the mock-up.

Next, the `<label>` links the displayed values, Yes and No, to their respective radio inputs. This link enhances accessibility by allowing users to click on the label's text as well as the radio buttons, which provides more clickable surface area.

PAUSE

There is an attribute on the `<label>` element called `for`. Explain the purpose of this attribute.

This is used to reference which radio button this label belongs to. The value for the `for` attribute must match the value of the `id` attribute of the `<input>` element.

Hide Answer

IMPORTANT

> The radio button elements must have the same `name` attribute in order to be associated with the same radio group. This is how the browser identifies that a group of choices can have an exclusive singular answer. In this case, the value of our radio group in `name` is `trainer-confirm`.

Add the Checkbox Input

Understanding what you do now about `<input>` element syntax and using Google if you need to, try to code the next checkbox on your own. For accessibility purposes, make sure to wrap the content of the `<p>` in the checkbox in a `<label>`, enabling the checkbox to toggle when anywhere in the content is clicked. This simple fix can greatly assist anyone with a motor-skills disability when trying to check a very small box.

The HTML for this is below, but don't peek! Try to do it yourself first.

Add the Submit Button

A button is an essential component of a user interface and can have a variety of different functions and uses. A button can link to another area on the webpage or an external site, submit data to a server, or be programmed for any activity.

Our button has a special function: to submit the user data gathered inside the `<form>`. Once again, we will configure our element with attributes to make it function according to our needs. In our case, we need a `submit` type to perform the needed action. The content of this element, "Get running", will be written inside the `<button>` element to render a label on the button itself.

Currently, our requirements in the mock-up don't have an action to assign to the button, but typically we would assign a URL path in the `<form>` element in the `action` attribute. To learn more, see the **MDN web docs on the form element (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form)**.

Let's take a look at our hero section's HTML code:

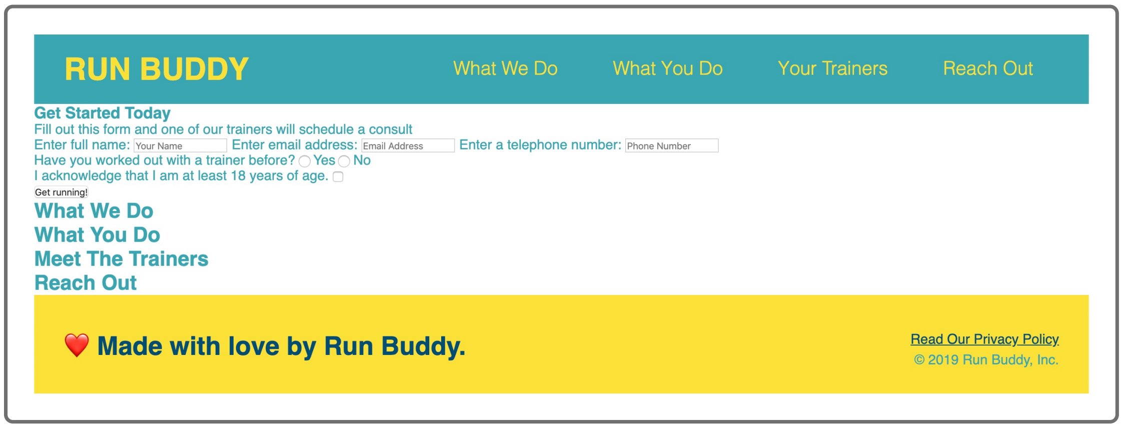
```
<!-- hero section -->
<section>
  <div>
    <h3>Get Started Today!</h3>
    <p>Fill out this form and one of our trainers will schedule a cons
    <form>
      <label for="name">Enter full name:</label>
      <input type="text" placeholder="Your Name" name="name" id="name"
      <label for="email">Enter email address:</label>
      <input type="text" placeholder="Email Address" name="email" id="
      <label for="phone">Enter a telephone number:</label>
      <input type="text" placeholder="Phone Number" name="phone" id="p
      <p>
        Have you worked out with a trainer before?
        <input type="radio" name="trainer-confirm" id="trainer-yes" />
        <label for="trainer-yes">Yes</label>
        <input type="radio" name="trainer-confirm" id="trainer-no" />
        <label for="trainer-no">No</label>
      </p>
      <p>
        <label for="checkbox" >
          I acknowledge that I am at least 18 years of age.
        </label>
        <input type="checkbox" name="age-confirm" id="checkbox" />
      </p>
      <button type="submit">
        Get running!
      </button>
    </form>
  </div>
</section>
<!-- end hero -->
```

HIDE PRO TIP

Notice that there are no empty lines within the `<section>` element. Being concise with language and syntax also pertains to extraneous lines and white space. Making sure there are no empty lines or missing indentation won't affect what's rendered in the browser, but these best practices enhance legibility and communicate parent/child relationships to other developers and even a future you, which can greatly improve development speed.

That was a lot of code we just entered! It's time to save and reload the page in the browser to see the current progress of our landing page. Make sure your file is saved, and then view it in the browser.

If your webpage resembles the following image, great job! If not, go back and audit your HTML until you get a result that resembles this image:



RUN BUDDY

What We Do What You Do Your Trainers Reach Out

Get Started Today

Fill out this form and one of our trainers will schedule a consult

Enter full name: Your Name Enter email address: Email Address Enter a telephone number: Phone Number

Have you worked out with a trainer before? Yes No

I acknowledge that I am at least 18 years of age.

Get running!

What We Do

What You Do

Meet The Trainers

Reach Out

Made with love by Run Buddy.

Read Our Privacy Policy

© 2019 Run Buddy, Inc.

Special Input Types

Our form is perfectly okay the way it is. But we can improve it by employing a couple of input types that were introduced in HTML5: `email` and `tel`. Browsers treat these input types differently than classic `text` input types.

When a user inputs a value into an `email` input type and clicks the submit button, the browser automatically validates that value. If the value doesn't look like an email address, the browser will not submit the form. Instead, the browser will display a helpful validation warning next to the `email` input field.

Desktop browsers treat the `tel` input type the same as the default `text` type. They don't automatically validate the user's inputted value. However, browsers on mobile devices will display a numeric keypad for the user to input a value, rather than a standard keyboard. This is a lovely experience for mobile users. Considering that the majority of website visits come from browsers on mobile devices, we should provide this experience.

It's easy to convert our email and phone inputs to these special types. Just change the `type` attribute of the email input to `email`, and the `type` attribute of the phone input to `tel`. Go ahead and do that in your `index.html` file now.

The HTML for these two inputs should now look like this:

```
<label for="email">Enter email address:</label>
<input type="email" placeholder="Email Address" name="email" id="email
<label for="phone">Enter a telephone number:</label>
<input type="tel" placeholder="Phone Number" name="phone" id="phone"/>
```

Save your work and refresh the page in the browser. Type some wacky non-email text into the email field and click the submit button. Can you get a validation warning to appear?

Testing the phone field is a little more complicated. First, you need to add, commit, and push your work to GitHub. Then you can grab a mobile device and load the project's GitHub Page in its browser. You should see a numeric keypad appear when you tap on the phone number field.

DEEP DIVE

DEEP DIVE

Browsers support a lot of input types. You can see a comprehensive list at **the MDN web docs on form input types (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#Form_%3Cinput%3E_types)**. Feel free to experiment with them, but keep in mind that many of them are not fully implemented in all browsers yet. If you click on a type in the MDN list and scroll to the very bottom of the page, you'll find a Browser Compatibility chart. This chart shows which browsers support the input type, as well as which features they support.

Get into the habit of researching how well browsers support new HTML and CSS before you use them.

Our foundation work is now complete so we can proceed to the design phase of this lesson after a quick assessment:

Final score: 100%
Retake