How to Make Custom Checkboxes in HTML

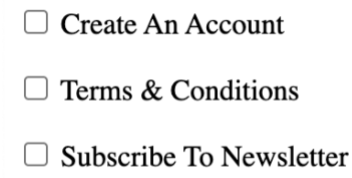
Written by Chad Jordan

4/17/22

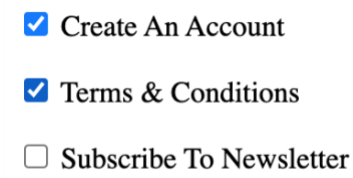
When creating forms on your website, it's important to make them as easy as pressing a button, or in this case, a check box! In this simple guide, we will be creating a custom set of checkboxes with animation effects, only using HTML and CSS, no JavaScript required! Whether your visitors are just verifying your terms and conditions, need to select multiple answers, or want to subscribe to your newsletter, we need to provide these options to them. This isn't just due to good HCI *(Human Computer Interaction)* practices for the visitors, but the importance of implementing good, and thorough functionality into your product's website.

Unlike radio buttons, checkboxes allow you to select multiple options/answers within a given category or multiple categories. These options make it possible for your visitors to perform multiple actions such as creating an account, agreeing to terms and conditions, and subscribing to a newsletter. Multiple checkboxes also allow you to get feedback from people who are visiting your site. This can work in your favor when placing meta tags in your code for SEO purposes.

Here's an example of simple checkboxes on a webpage with no CSS applied:



As it stands, we can click the boxes and it will display a checkbox.



With these explanations and examples of using checkboxes, let's cover more details about creating them in HTML.

HTML Checkbox Input Tag

In HTML, a checkbox is an **<input>** tag with "checkbox" assigned to the attribute type. The syntax is represented as:

```
<input type="checkbox">
```

Just as the above example displays, even this syntax with no CSS applied will create checkboxes on your site, but in this guide, we want to customize our checkboxes to provide more style and match the content and flow of our website.

Let's begin the implementation of our custom checkboxes with some basic code examples below.

HTML Code Example

In order to create the checkboxes that you saw in the above examples, the following syntax is how we would implement them in HTML:

```
<div class="container">>
   <form>
       <p>
         <input type="checkbox" id="Account" name="checkboxes">
         <label for="Account">Create An Account</label>
       </p>
       <p>
         <input type="checkbox" id="Terms" name="checkboxes">
         <label for="Terms">Terms & Conditions</label>
       </p>
       <p>
         <input type="checkbox" id="Newsletter"
name="checkboxes">
         <label for="Newsletter">Subscribe To Newsletter</label>
       </p>
   </form>
</div>
```

We create a ** ** called 'container' and essentially nested tags inside of our paragraph tags as this is a very simple way of reading and structuring our code. We can have different names for the checkboxes via the **<label>** tag, but the name attribute, 'checkboxes' must remain static.

CSS Container Setup

The default placement, appearance, and click behavior will work, but it does not follow a customized look and behavior of my choice. This is where we utilize CSS to create and customize our container for our checkbox the way we want it to look. Since we aren't writing any JavaScript for this demo, we'll be writing CSS elements that are referred to as data selectors. In this case the **<input>** tag holds the attribute, 'checkbox' which will have the selector data set to it for all appearance and animated behaviors of our checkboxes.

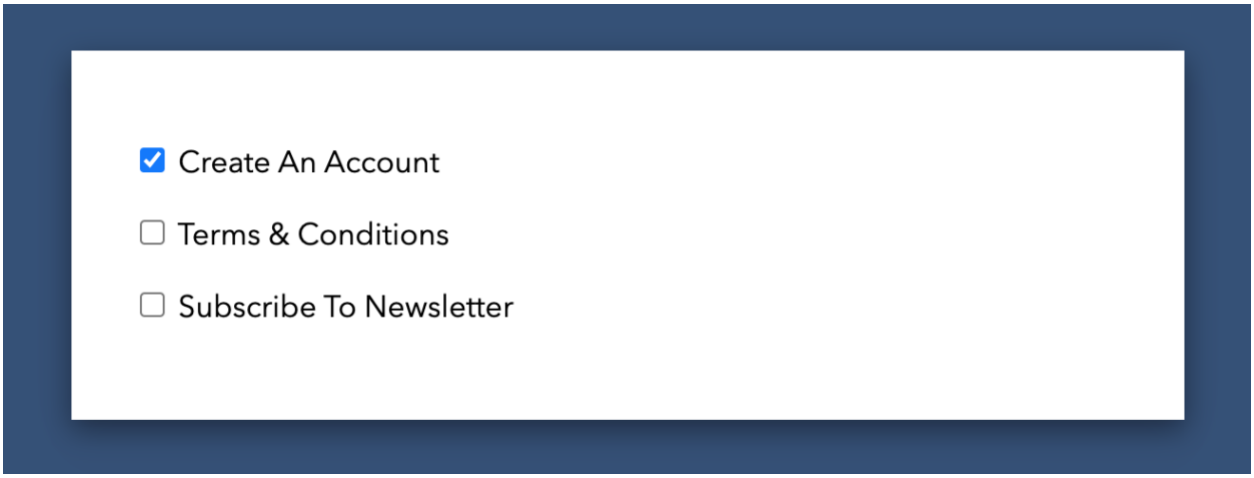
We begin by creating the style for the body of our container:

```
body {
  display: flex;
  justify-content: center;
  align-items: center;
```

```
  height: 100vh;
  font-family: "Avenir Next", san-serif;
  background-color: #345078;
}

.container {
  margin-bottom: 40vh;
  width: 65vw;
  box-shadow: 0 15px 35px rgba(50,50,93,.1),0 5px 15px
rgba(0,0,0,.45);
  padding: 2em;
  background-color: #fff;
}
```

The result will look like this:



CSS Checkbox Style

As mentioned above, this guide won't be using JavaScript, but instead writing some custom CSS selectors that affect the 'checkbox' data element that is referenced in our HTML. Browser support is not always good using rules for form elements. Instead, we're going to hide the checkbox input fields and then redraw them using the pseudo 'before' and 'after' elements.

This can be done in a few different ways, but we can just set the opacity to 0 so the checkboxes can still remain in the container, they are just not visible.

```
[type="checkbox"] {
  opacity: 0;
}
```

The next step involves copying the existing selector and creating another one and pasting it underneath. We will provide 'before' and 'after' data for the 'label' tag when the checkboxes are clicked.

```
[type="checkbox"] + label {
  position: relative;
  padding-left: 30px;
  cursor: pointer;
  display: inline-black;
  color: #888;
  line height: 25px;
}

[type="checkbox"] + label::before {
  content: "";
  position: absolute;
  left: 0;
  top: 0;
  width: 18px;
  height: 18px;
  outline: 2px solid #bbb;
  background: #fff;
}
```

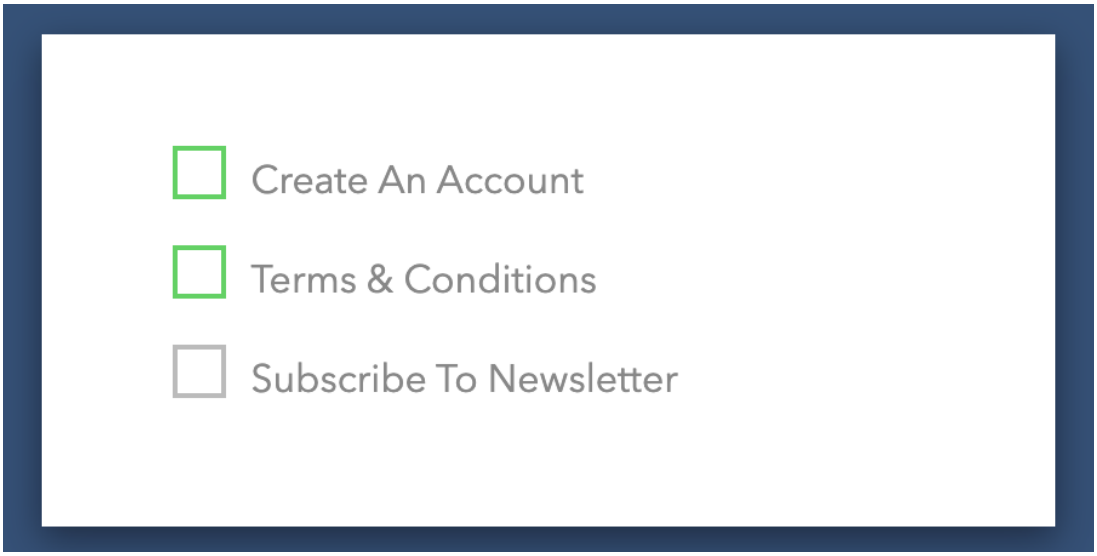
The 'before' element will be the box around the check box and the 'after' element will be the behavior of the actual checkmark.

The process of this next selector will check if the box is in its checked state, and then select the next adjacent label and modify its 'before' pseudo-element.

```
[type="checkbox"]checked + label::before {
  content: "";
  position: absolute;
  left: 0;
  top: 0;
```

```
  width: 18px;
  height: 18px;
  outline: 2px solid #5fd25f;
  background: #fff;
}
```

With this code in place, we can see in the following screenshot that it's doing exactly what it's supposed to do.

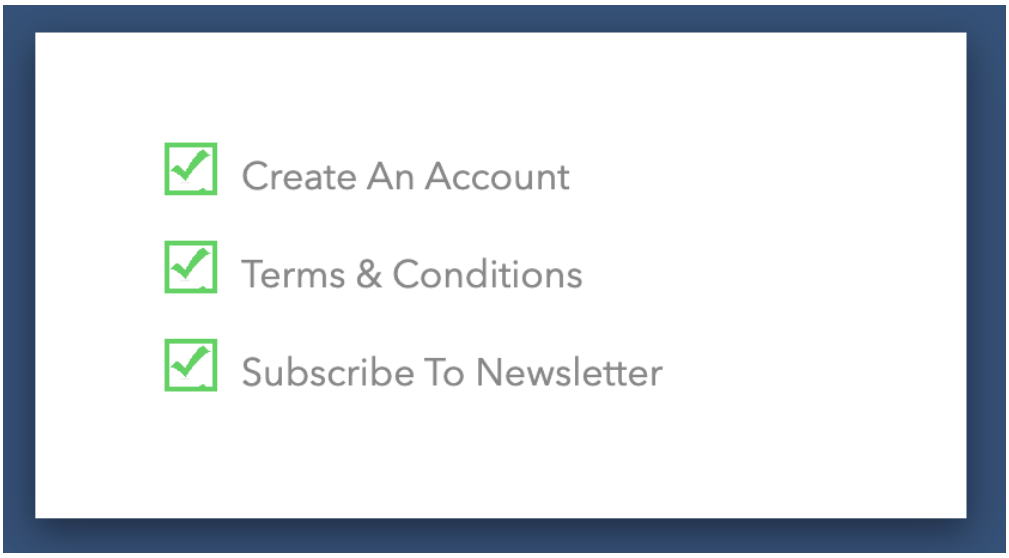


For this next portion of the guide, you'll need to either create or download a checkmark icon that you can animate for the checkbox. We're using a checkmark that was downloaded from **Wikimedia Commons**.

At this stage of the guide, we've prepared the behavior of our 'before' element, but now we want to address the 'after' element.

```
[type="checkbox"]:checked + label::after {
  content: "";
  position: absolute;
  left: 0;
  top: 0;
  width: 18px;
  height: 18px;
  background-image: url(checkmark.png);
  background-size: contain;
}
```

With this selector in place, our green checkmark now properly shows up in our checkbox:



It looks good, but with a few more lines of CSS, we can easily animate the checkmark to make it all the more visually pleasing to the eye. This last section affects any boxes that are currently not checked and looks for the next adjacent label, and then styles the 'after' pseudo-element.

```
[type="checkbox"]:not(:checked) + label::after {
  content: "";
  position: absolute;
  left: 0;
  top: 0;
  width: 18px;
  height: 18px;
  background-image: url(checkmark.png);
  background-size: contain;
  transform: scale(0);
  opacity: 0;
  transition: all .3s ease;
}
```

Once clicked, the checkmark scales up with a smooth animation effect, and then when unchecked, it smoothly scales back and fades out. Would you like to try the fully responsive demo? Click **here**. This concludes the guide on how to make your own custom checkbox with HTML and CSS.

Resources Used:

Love2dev.com

commons.wikimedia.org