

# How to style a checkbox using CSS

Asked 14 years, 1 month ago   Modified 8 days ago   Viewed 2.4m times



1087



I am trying to style a checkbox using the following:

```
<input type="checkbox" style="border:2px dotted #00f;display:block;background:#ff0000;" />
```



Run code snippet

[Expand snippet](#)

But the style is not applied. The checkbox still displays its default style. How do I give it the specified style?

html

css

checkbox

Share

Improve this question

Follow

edited Jul 17, 2019 at 15:35



Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Nov 10, 2010 at 19:57



Salman Virk

12.3k ● 9 ● 38 ● 47

[github.com/avatec/avatec-bootstrap3-custom-checkbox](https://github.com/avatec/avatec-bootstrap3-custom-checkbox) ready to use plugin

– Grzegorz Miśkiewicz Feb 13, 2020 at 14:04

7 Whatever decision you make for styling checkboxes or radio buttons via CSS please make sure that they are accessible. As of this comment I believe only 2 of the 33 answers so far are accessible. For the rest of the answers you're cutting off most, if not all accessibility. ( `ctrl+f` "accessibility") – maxshuty Jul 4, 2020 at 16:40 ✎

7 There's a native CSS property for this now, skip to [this answer](#). – TylerH Sep 13, 2021 at 14:46

Along with the `accent-color` CSS that (in most implementations) affects only the color when checked, it's also possible to further customise the background color when unchecked by using CSS `filters`, see [here](#) (very useful for dark mode styles where the white of empty checkboxes don't fit in). – Silveri Nov 1, 2022 at 16:09

44 Answers

Sorted by: Highest score (default)



1

2

Next



You can achieve quite a cool custom checkbox effect by using the new abilities that come with the `:after` and `:before` pseudo classes. The advantage to this, is: You

177 don't need to add anything more to the DOM, just the standard checkbox.



+50



Note this will only work for compatible browsers. I believe this is related to the fact that some browsers do not allow you to set `:after` and `:before` on input elements. Which unfortunately means for the moment only WebKit browsers are supported. Firefox + Internet Explorer will still allow the checkboxes to function, just unstyled, and this will hopefully change in the future (the code does not use vendor prefixes).

**This is a WebKit browser solution only (Chrome, Safari, Mobile browsers)**

[See Example Fiddle](#)

```
$(function() {  
  $('input').change(function() {  
    $('div').html(Math.random());  
  });  
});
```

```
/* Main Classes */  
.myinput[type="checkbox"]:before {  
  position: relative;  
  display: block;  
  width: 11px;  
  height: 11px;  
  border: 1px solid #808080;  
  content: "";  
  background: #FFF;  
}  
  
.myinput[type="checkbox"]:after {  
  position: relative;  
  display: block;  
  left: 2px;  
  top: -11px;  
  width: 7px;  
  height: 7px;  
  border-width: 1px;  
  border-style: solid;  
  border-color: #B3B3B3 #dcddde #dcddde #B3B3B3;  
  content: "";  
  background-image: linear-gradient(135deg, #B1B6BE 0%, #FFF 100%);  
  background-repeat: no-repeat;  
  background-position: center;  
}  
  
.myinput[type="checkbox"]:checked:after {  
  background-image:  
url('data:image/png;base64,iVBORw0KGgoAAAANSUheUgAAAcAAAAHCAQAAABuW59YAAAACXBIW  
linear-gradient(135deg, #B1B6BE 0%, #FFF 100%);  
}  
  
.myinput[type="checkbox"]:disabled:after {  
  -webkit-filter: opacity(0.4);  
}  
  
.myinput[type="checkbox"]:not(:disabled):checked:hover:after {
```

```

    background-image:
url('data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAACAAAAHCAQAAABuW59YAAAACXBIW
linear-gradient(135deg, #8BB0C2 0%, #FFF 100%);
}

.myinput[type="checkbox"]:not(:disabled):hover:after {
    background-image: linear-gradient(135deg, #8BB0C2 0%, #FFF 100%);
    border-color: #85A9BB #92C2DA #92C2DA #85A9BB;
}

.myinput[type="checkbox"]:not(:disabled):hover:before {
    border-color: #3D7591;
}

/* Large checkboxes */
.myinput.large {
    height: 22px;
    width: 22px;
}

.myinput.large[type="checkbox"]:before {
    width: 20px;
    height: 20px;
}

.myinput.large[type="checkbox"]:after {
    top: -20px;
    width: 16px;
    height: 16px;
}

/* Custom checkbox */
.myinput.large.custom[type="checkbox"]:checked:after {
    background-image:
url('data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAAAAAf8/9hAAAAGHRFW
linear-gradient(135deg, #B1B6BE 0%, #FFF 100%);
}

.myinput.large.custom[type="checkbox"]:not(:disabled):checked:hover:after {
    background-image:
url('data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAAAAAf8/9hAAAAGHRFW
linear-gradient(135deg, #8BB0C2 0%, #FFF 100%);
}

```

```

<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js">
</script>

```

```

<table style="width:100%">
  <tr>
    <td>Normal:</td>
    <td><input type="checkbox" /></td>
    <td><input type="checkbox" checked="checked" /></td>
    <td><input type="checkbox" disabled="disabled" /></td>
    <td><input type="checkbox" disabled="disabled" checked="checked" /></td>
  </tr>
  <tr>
    <td>Small:</td>
    <td><input type="checkbox" class="myinput" /></td>
    <td><input type="checkbox" checked="checked" class="myinput" /></td>
    <td><input type="checkbox" disabled="disabled" class="myinput" /></td>
    <td><input type="checkbox" disabled="disabled" checked="checked"

```

```

class="myinput" /></td>
</tr>
<tr>
  <td>Large:</td>
  <td><input type="checkbox" class="myinput large" /></td>
  <td><input type="checkbox" checked="checked" class="myinput large" /></td>
  <td><input type="checkbox" disabled="disabled" class="myinput large" />
</td>
  <td><input type="checkbox" disabled="disabled" checked="checked"
class="myinput large" /></td>
</tr>
<tr>
  <td>Custom icon:</td>
  <td><input type="checkbox" class="myinput large custom" /></td>
  <td><input type="checkbox" checked="checked" class="myinput large custom"
/></td>
  <td><input type="checkbox" disabled="disabled" class="myinput large custom"
/></td>
  <td><input type="checkbox" disabled="disabled" checked="checked"
class="myinput large custom" /></td>
</tr>
</table>

```



Run code snippet

[Expand snippet](#)

## [Bonus Webkit style flipswitch fiddle](#)

```

$(function() {
  var f = function() {
    $(this).next().text($(this).is(':checked') ? ':checked' :
':not(:checked)');
  };
  $('input').change(f).trigger('change');
});

```

```

body {
  font-family: arial;
}

.flipswitch {
  position: relative;
  background: white;
  width: 120px;
  height: 40px;
  -webkit-appearance: initial;
  border-radius: 3px;
  -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
  outline: none;
  font-size: 14px;
  font-family: Trebuchet, Arial, sans-serif;
  font-weight: bold;
  cursor: pointer;
  border: 1px solid #ddd;
}

.flipswitch:after {

```

```

position: absolute;
top: 5%;
display: block;
line-height: 32px;
width: 45%;
height: 90%;
background: #fff;
box-sizing: border-box;
text-align: center;
transition: all 0.3s ease-in 0s;
color: black;
border: #888 1px solid;
border-radius: 3px;
}

.flipswitch:after {
  left: 2%;
  content: "OFF";
}

.flipswitch:checked:after {
  left: 53%;
  content: "ON";
}

```

```

<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>

<h2>Webkit friendly mobile-style checkbox/flipswitch</h2>
<input type="checkbox" class="flipswitch" /> &nbsp;
<span></span>
<br>
<input type="checkbox" checked="checked" class="flipswitch" /> &nbsp;
<span></span>

```

▶ Run code snippet

🔗 [Expand snippet](#)

Share Improve this answer

Follow

edited Jul 17, 2019 at 17:52



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Jun 9, 2013 at 1:16



Josh Mc

10.2k ● 8 ● 56 ● 68

13 Firefox 33.1/Linux: The fiddle shows just default checkboxes. Nothing looks different. – [robsch](#) Nov 27, 2014 at 7:33

3 @robsch This is clearly pointed out in the original post. Version of firefox or OS is irrelevant, it does not work in firefox. "FF + IE will still allow the check-boxes to function, just un-styled..." – [Josh Mc](#) Jan 4, 2015 at 22:25 ✎

A good approach. But not all browsers doing it good. Only Chrome has the best output as far as I examined. – [E-A](#) Jun 9, 2015 at 12:56

Very nice. However... It is again an IE 5.x approach. Webkit only. Because it doesn't always follow the rules... That's the entire problem with webkit browsers. – [Alex](#) Mar 23, 2016 at 11:17

- 1 i believe this is invalid and webkit/blink are violating the spec. `::before` and `::after` only work on containers; checkboxes are replaced elements. same reason you can't use them on images. – [Eevee](#) Dec 8, 2019 at 3:21
- 



## Before you begin (as of Jan 2015)

171

The original question and answer are now ~5 years old. As such, this is a little bit of an update.



Firstly, there are a number of approaches when it comes to styling checkboxes. The basic tenet is:



1. You will need to hide the default checkbox control which is styled by your browser, and cannot be overridden in any meaningful way using CSS.
2. With the control hidden, you will still need to be able to detect and toggle its checked state.
3. The checked state of the checkbox will need to be reflected by styling a new element.

## The solution (in principle)

The above can be accomplished by a number of means — and you will often hear that using CSS3 pseudo-elements is the right way. Actually, there is no real right or wrong way, it depends on the approach most suitable for the context you will be using it in. That said, I have a preferred one.

1. Wrap your checkbox in a `label` element. This will mean that even when it is hidden, you can still toggle its checked state by clicking anywhere within the label.
2. Hide your checkbox.
3. Add a new element *after* the checkbox which you will style accordingly. It must appear after the checkbox so it can be selected using CSS and styled dependent on the `:checked` state. CSS cannot select 'backwards'.

## The solution (in code)

► [Show code snippet](#)

## Refinement (using icons)

"But hey!" I hear you shout. What about if I want to show a nice little tick or cross in the box? And I don't want to use background images!

Well, this is where CSS3's pseudo-elements can come into play. These support the `content` property which allows you to inject [Unicode icons](#) representing either state. Alternatively, you could use a third party font icon source such as font awesome (though make sure you also set the relevant `font-family`, e.g. to `FontAwesome` )

```
label input {
  display: none; /* Hide the default checkbox */
}

/* Style the artificial checkbox */
label span {
  height: 10px;
  width: 10px;
  border: 1px solid grey;
  display: inline-block;
  position: relative;
}

/* Style its checked state...with a ticked icon */
[type=checkbox]:checked + span:before {
  content: '\2714';
  position: absolute;
  top: -5px;
  left: 0;
}
```

```
<label>
  <input type='checkbox'>
  <span></span>
  Checkbox label text
</label>
```

[Run code snippet](#)[Expand snippet](#)

Share Improve this answer

Follow

edited Apr 18 at 21:37



Юлиан Антонов

19 ● 2

answered Jan 5, 2015 at 11:59



SW4

71.1k ● 20 ● 136 ● 139

15 Except for: simplified HTML, simplified CSS, detailed explanation. – [SW4](#) Jan 30, 2015 at 8:47

4 @AnthonyHayward - updated answer, this was due to using display:none which does not instantiate the control in a tabbable way, I've changed – [SW4](#) Oct 8, 2015 at 21:02

2 I was struggling to find an example in which the checkbox was inside of the label rather than before/after it. This is a very well documented and explained solution. It would be great to see this answer updated for January 2016. – [Clarus Dignus](#) Jan 16, 2016 at 7:19 ✎

1 Still not tabbable with `display: none`. – [sam](#) Mar 4, 2016 at 17:29

2 Replace `visibility: hidden;` with `opacity: 0 !important;` if you're still having trouble with tabbing. – [jabacchetta](#) Feb 8, 2017 at 7:32 ✎



153



There is a way to do this using just CSS. We can (ab)use the `label` element and style that element instead. The caveat is that this will not work for Internet Explorer 8 and lower versions.

```
.myCheckbox input {
  position: relative;
  z-index: -9999;
}

.myCheckbox span {
  width: 20px;
  height: 20px;
  display: block;
  background: url("link_to_image");
}

.myCheckbox input:checked + span {
  background: url("link_to_another_image");
}
```

```
<label for="test">Label for my styled "checkbox"</label>
<label class="myCheckbox">
  <input type="checkbox" name="test" />
  <span></span>
</label>
```

▶ Run code snippet

🔗 [Expand snippet](#)

Share Improve this answer

Follow

edited Jul 17, 2019 at 17:47



[Peter Mortensen](#)

31.6k ● 22 ● 109 ● 133

answered Aug 30, 2012 at 8:59



[Blake Pettersson](#)

9,067 ● 3 ● 28 ● 36

@GandalfStormCrow this will work for any browser that supports the `:checked` pseudo-class, which IE8 does NOT support. You can check if this works with [selectivizr.com](#) - which adds



support for :checked and friends. – [Blake Pettersson](#) Oct 5, 2012 at 12:08

In other words, IE9 and later versions supports :checked. – [Blake Pettersson](#) Oct 8, 2012 at 13:13

- 1 There is a polyfill for IE8 and below: [github.com/rdebeasi/checked-polyfill](https://github.com/rdebeasi/checked-polyfill) – [Tim](#) Jan 2, 2014 at 19:49

It's working, but first time it flickers. Why is it happening? – [technophyle](#) Mar 4, 2016 at 2:26

I believe hidden `input` s never take keyboard focus, so these are unreachable by keyboard.  
– [sam](#) Mar 4, 2016 at 17:27



134



## Modern *accessible* solution - use `accent-color`

Use the new `accent-color` property and make certain to meet a proper [contrast ratio](#) of 3:1 to ensure [accessibility](#). This also works for `radio` buttons.

```
.red-input {  
  accent-color: #9d3039;  
  height: 20px; /* not needed */  
  width: 20px; /* not needed */  
}
```

```
<input class="red-input" type="checkbox" />  
  
<!-- Radio button example -->  
<input class="red-input" type="radio" />
```



Run code snippet

[Expand snippet](#)

## Old answer, I only recommend this if you need more customization than the above offers:

I have been scrolling and scrolling and **tons** of these answers simply throw accessibility out the door and [violate WCAG](#) in more than one way. I threw in radio buttons since most of the time when you're using custom checkboxes you want custom radio buttons too.

### Fiddles:

- [Checkboxes](#) - pure CSS - free from 3rd party libraries
- [Radio buttons](#) - pure CSS - free from 3rd party libraries

- [Checkboxes\\*](#) that use FontAwesome but could be swapped with Glyphicons, etc. easily

Late to the party but somehow this is still difficult in ~~2019~~, ~~2020~~, 2021 so I have added my three solutions which are **accessible** and **easy** to drop in.

These are all **JavaScript free**, **accessible**, and **external library free**\*...

If you want to plug-n-play with any of these just copy the style sheet from the fiddles, edit the color codes in the CSS to fit your needs, and be on your way. You can add a custom svg checkmark icon if you want for the checkboxes. I've added lots of comments for those non-CSS'y folks.

If you have long text or a small container and are encountering text wrapping underneath the checkbox or radio button input then just convert to divs [like this](#).

**Longer explanation:** I needed a solution that does not violate WCAG, doesn't rely on JavaScript or external libraries, and that does not break keyboard navigation like tabbing or spacebar to select, that allows focus events, a solution that allows for disabled checkboxes that are both checked and unchecked, and finally a solution where I can customize the look of the checkbox however I want with different `background-color`'s, `border-radius`, `svg` backgrounds, etc.

I used some combination of [this answer](#) from @Jan Turoň to come up with my own solution which seems to work quite well. I've done a radio button fiddle that uses a lot of the same code from the checkboxes in order to make this work with radio buttons too.

I am still learning accessibility so if I missed something please drop a comment and I will try to correct it.

Here is a code example of my checkboxes:

```
input[type="checkbox"] {
  position: absolute;
  opacity: 0;
  z-index: -1;
}

/* Text color for the label */
input[type="checkbox"]+span {
  cursor: pointer;
  font: 16px sans-serif;
  color: black;
}

/* Checkbox un-checked style */
input[type="checkbox"]+span:before {
  content: '';
  border: 1px solid grey;
  border-radius: 3px;
  display: inline-block;
```

```

width: 16px;
height: 16px;
margin-right: 0.5em;
margin-top: 0.5em;
vertical-align: -2px;
}

/* Checked checkbox style (in this case the background is green #e7ffba, change
this to change the color) */
input[type="checkbox"]:checked+span:before {
  /* NOTE: Replace the url with a path to an SVG of a checkmark to get a
checkmark icon */
  background-image:
url('https://cdnjs.cloudflare.com/ajax/libs/ionicons/4.5.6/collection/build/ionicons/ionicons.svg');
  background-repeat: no-repeat;
  background-position: center;
  /* The size of the checkmark icon, you may/may not need this */
  background-size: 25px;
  border-radius: 2px;
  background-color: #e7ffba;
  color: white;
}

/* Adding a dotted border around the active tabbed-into checkbox */
input[type="checkbox"]:focus+span:before,
input[type="checkbox"]:not(:disabled)+span:after {
  /* Visible in the full-color space */
  box-shadow: 0px 0px 0px 2px rgba(0, 150, 255, 1);

  /* Visible in Windows high-contrast themes
  box-shadow will be hidden in these modes and
  transparency will not be hidden in high-contrast
  thus box-shadow will not show but the outline will
  providing accessibility */
  outline-color: transparent; /*switch to transparent*/
  outline-width: 2px;
  outline-style: dotted;
}

/* Disabled checkbox styles */
input[type="checkbox"]:disabled+span {
  cursor: default;
  color: black;
  opacity: 0.5;
}

/* Styles specific to this fiddle that you do not need */
body {
  padding: 1em;
}
h1 {
  font-size: 18px;
}

```

<h1>

NOTE: Replace the url for the background-image in CSS with a path to an SVG in your solution or CDN. This one was found from a quick google search for a checkmark icon cdn

</h1>

<p>You can easily change the background color, checkbox symbol, border-radius, etc.</p>

```
<label>
  <input type="checkbox">
  <span>Try using tab and space</span>
</label>
```

```
<br>
```

```
<label>
  <input type="checkbox" checked disabled>
  <span>Disabled Checked Checkbox</span>
</label>
```

```
<br>
```

```
<label>
  <input type="checkbox" disabled>
  <span>Disabled Checkbox</span>
</label>
<br>
```

```
<label>
  <input type="checkbox">
  <span>Normal Checkbox</span>
</label>
```

```
<br>
```

```
<label>
  <input type="checkbox">
  <span>Another Normal Checkbox</span>
</label>
```



Run code snippet

[Expand snippet](#)

Share Improve this answer

edited Feb 8, 2023 at 20:41

answered Oct 26, 2019 at 12:39

Follow



maxshuty

10.6k ● 13 ● 69 ● 85

It's one thing to update the answer to provide a new solution, although kinda iffy when another answer already provides that solution. It's another thing altogether to delete your existing content and show only the new solution that is already entirely provided by a separate answer. If all that's left is the new solution copying another answer, then the post should just be deleted.

– TylerH Aug 17, 2022 at 15:39



I always use pseudo elements `:before` and `:after` for changing the appearance of checkboxes and radio buttons. it's works like a charm.

108

Refer this [link](#) for more info



## Steps

1. Hide the default checkbox using css rules like `visibility:hidden` or `opacity:0` or `position:absolute;left:-9999px` etc.
2. Create a fake checkbox using `:before` element and pass either an empty or a non-breaking space `'\00a0'`;
3. When the checkbox is in `:checked` state, pass the unicode `content: "\2713"`, which is a checkmark;
4. Add `:focus` style to make the checkbox accessible.
5. Done

Here is how I did it.

```
.box {
  background: #666666;
  color: #ffffff;
  width: 250px;
  padding: 10px;
  margin: 1em auto;
}
p {
  margin: 1.5em 0;
  padding: 0;
}
input[type="checkbox"] {
  visibility: hidden;
}
label {
  cursor: pointer;
}
input[type="checkbox"] + label:before {
  border: 1px solid #333;
  content: "\00a0";
  display: inline-block;
  font: 16px/1em sans-serif;
  height: 16px;
  margin: 0 .25em 0 0;
  padding: 0;
  vertical-align: top;
  width: 16px;
}
input[type="checkbox"]:checked + label:before {
  background: #fff;
  color: #333;
  content: "\2713";
  text-align: center;
}
input[type="checkbox"]:checked + label:after {
  font-weight: bold;
}
input[type="checkbox"]:focus + label::before {
```

```
outline: rgb(59, 153, 252) auto 5px;
}
```

```
<div class="content">
  <div class="box">
    <p>
      <input type="checkbox" id="c1" name="cb">
      <label for="c1">Option 01</label>
    </p>
    <p>
      <input type="checkbox" id="c2" name="cb">
      <label for="c2">Option 02</label>
    </p>
    <p>
      <input type="checkbox" id="c3" name="cb">
      <label for="c3">Option 03</label>
    </p>
  </div>
</div>
```



Run code snippet

[Expand snippet](#)

Much more stylish using `:before` and `:after`

```
body{
  font-family: sans-serif;
}

.container {
  margin-top: 50px;
  margin-left: 20px;
  margin-right: 20px;
}

.checkbox {
  width: 100%;
  margin: 15px auto;
  position: relative;
  display: block;
}

.checkbox input[type="checkbox"] {
  width: auto;
  opacity: 0.00000001;
  position: absolute;
  left: 0;
  margin-left: -20px;
}

.checkbox label {
  position: relative;
}

.checkbox label:before {
  content: '';
  position: absolute;
  left: 0;
  top: 0;
  margin: 4px;
```

```

width: 22px;
height: 22px;
transition: transform 0.28s ease;
border-radius: 3px;
border: 2px solid #7bbe72;
}
.checkbox label:after {
  content: '';
  display: block;
  width: 10px;
  height: 5px;
  border-bottom: 2px solid #7bbe72;
  border-left: 2px solid #7bbe72;
  -webkit-transform: rotate(-45deg) scale(0);
  transform: rotate(-45deg) scale(0);
  transition: transform ease 0.25s;
  will-change: transform;
  position: absolute;
  top: 12px;
  left: 10px;
}
.checkbox input[type="checkbox"]:checked ~ label::before {
  color: #7bbe72;
}

.checkbox input[type="checkbox"]:checked ~ label::after {
  -webkit-transform: rotate(-45deg) scale(1);
  transform: rotate(-45deg) scale(1);
}

.checkbox label {
  min-height: 34px;
  display: block;
  padding-left: 40px;
  margin-bottom: 0;
  font-weight: normal;
  cursor: pointer;
  vertical-align: sub;
}
.checkbox label span {
  position: absolute;
  top: 50%;
  -webkit-transform: translateY(-50%);
  transform: translateY(-50%);
}
.checkbox input[type="checkbox"]:focus + label::before {
  outline: 0;
}

```

```

<div class="container">
  <div class="checkbox">
    <input type="checkbox" id="checkbox" name="" value="">
    <label for="checkbox"><span>Checkbox</span></label>
  </div>

  <div class="checkbox">
    <input type="checkbox" id="checkbox2" name="" value="">
    <label for="checkbox2"><span>Checkbox</span></label>
  </div>
</div>

```

[Run code snippet](#)[Expand snippet](#)[Share](#) [Improve this answer](#)

edited Feb 3, 2020 at 4:36

answered Dec 21, 2015 at 4:50

[Follow](#)**Jinu Kurian**

9,396 ● 3 ● 29 ● 36

**61**

I'd follow the advice of [SW4's answer](#). Not anymore: [Volomike's answer](#) is far superior to all the answers here (note my suggested improvement in the comment to the answer). Proceed reading this answer if you are curious about alternative approaches, which this answer comments.

First of all, hide the checkbox and to cover it with a custom span, suggesting this HTML:

```
<label>
  <input type="checkbox">
  <span>send newsletter</span>
</label>
```

The wrap in label neatly allows clicking the text without the need of "for-id" attribute linking. However,

## Do not hide it using `visibility: hidden` or `display: none`

It works by clicking or tapping, but that is a lame way to use checkboxes. Some people still use much more effective `Tab` to move focus, `Space` to activate, and hiding with that method disables it. If the form is long, one will save someone's wrists to use `tabindex` or `accesskey` attributes. And if you observe the system checkbox behavior, there is a decent shadow on hover. The well styled checkbox should follow this behavior.

[cobberboy's answer](#) recommends [Font Awesome](#) which is usually better than bitmap since fonts are scalable vectors. Working with the HTML above, I'd suggest these CSS rules:

### 1. Hide checkboxes

```
input[type="checkbox"] {
  position: absolute;
  opacity: 0;
  z-index: -1;
}
```

I use just negative `z-index` since my example uses big enough checkbox skin to cover it fully. I don't recommend `left: -999px` since it is not reusable in every



layout. [Bushan wagh's answer](#) provides a bulletproof way to hide it and convince the browser to use tabindex, so it is a good alternative. Anyway, both is just a hack. The proper way today is `appearance: none`, see [Joost's answer](#):

```
input[type="checkbox"] {  
  appearance: none;  
  -webkit-appearance: none;  
  -moz-appearance: none;  
}
```

## 2. Style checkbox label

```
input[type="checkbox"] + span {  
  font: 16pt sans-serif;  
  color: #000;  
}
```

## 3. Add checkbox skin

```
input[type="checkbox"] + span:before {  
  font: 16pt FontAwesome;  
  content: '\00f096';  
  display: inline-block;  
  width: 16pt;  
  padding: 2px 0 0 3px;  
  margin-right: 0.5em;  
}
```

`\00f096` is Font Awesome's `square-o`, padding is adjusted to provide even dotted outline on focus (see below).

## 4. Add checkbox checked skin

```
input[type="checkbox"]:checked + span:before {  
  content: '\00f046';  
}
```

`\00f046` is Font Awesome's `check-square-o`, which is not the same width as `square-o`, which is the reason for the width style above.

## 5. Add focus outline

```
input[type="checkbox"]:focus + span:before {  
  outline: 1px dotted #aaa;  
}
```

Safari doesn't provide this feature (see @Jason Sankey's comment), see [this answer](#) for Safari-only CSS

## 6. Set gray color for disabled checkbox

```
input[type="checkbox"]:disabled + span {  
  color: #999;  
}
```

```
}
```

## 7. Set hover shadow on non-disabled checkbox

```
input[type="checkbox"]:not(:disabled) + span:hover:before {  
  text-shadow: 0 1px 2px #77F;  
}
```

## Test it on [JS Fiddle](#)

Try to hover the mouse over the checkboxes and use Tab and Shift + Tab to move and Space to toggle.

Share Improve this answer

edited Aug 29, 2021 at 8:28

answered Sep 30, 2015 at 21:43

Follow



Jan Turoň

32.8k ● 23 ● 137 ● 177

- 3 Upvote for covering the focus issue: customisations like this shouldn't remove basic functionality. Note though that Safari doesn't give keyboard focus to checkboxes (even default ones), that confused me momentarily when implementing a solution that included focus handling.  
– [Jason Sankey](#) Feb 15, 2016 at 2:54

I ended up doing a spin off of this answer in mine. I could not rely upon external libraries like FontAwesome, so I baked some advice from this answer into my own custom solution and kept everything accessible like this answer does. I also included a radio button design:  
[stackoverflow.com/a/58570835/4826740](https://stackoverflow.com/a/58570835/4826740) Thanks Jan for the awesome help! – [maxshuty](#) May 16, 2020 at 13:36



54



The [CSS Basic User Interface Module Level 4](#) adds support for this (finally) via a new solution called `accent-color`, and it's *actually* quite simple, unlike pretty much every other answer here:

```
input {  
  accent-color: rebeccapurple;  
}
```

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



Run code snippet

[Expand snippet](#)

Set whatever CSS color (e.g. named value, hex code, etc.) you want as the value of `accent-color`, and it will be applied.

This currently works in Chrome (v93+), Edge (v93+), Firefox (v92+), Opera (v79+), and Safari (v15.4+).

Note: Edge, Chrome, and Opera (and possibly Safari; I can't test that) currently don't support alpha channel values via `rgba()` either (the RGB values of `rgba()` will still "work"; the alpha channel will simply be ignored by the browser). See [MDN Browser Support](#) for more information. See also [Chromium Bug Tracking](#) for the status of support in Chromium browsers.

Share Improve this answer

edited Dec 13 at 14:55

answered Sep 13, 2021 at 14:46

Follow



TylerH

21.2k ● 76 ● 79 ● 110

It seems like Chromium's lack of alpha support is intentional; their implementation of `accent-color` renders the entire checkbox invisible with an alpha transparency value of zero. They're re-investigating it and looking into copying Firefox' method as a fix to allow alpha support: [bugs.chromium.org/p/chromium/issues/detail?id=1200932](https://bugs.chromium.org/p/chromium/issues/detail?id=1200932) – TylerH Aug 19, 2022 at 16:03

`accent-color` works until the checkbox is disabled, then colors revert to browser defaults (unless there's some other CSS attribute?) Confirmed with MS Edge Version 120.0.2210.91, probably also in modern Chrome. – BobHy Jan 5 at 1:30

@BobHy `accent-color` only applies to the *checked* format of a checkbox, so how exactly would you expect it to even be possible for `disabled`? In other words, of course it doesn't work when the checkbox is disabled! The `disabled` property also applies its own styles, I believe, so you could potentially override those, but I haven't looked into that. – TylerH Jan 5 at 15:17



52



With pure CSS, nothing fancy with `:before` and `:after`, no transforms, you can turn off the default appearance and then style it with an inline background image like the following example. This works in Chrome, Firefox, Safari, and now Edge (Chromium Edge).

```
INPUT[type=checkbox]:focus
{
    outline: 1px solid rgba(0, 0, 0, 0.2);
}

INPUT[type=checkbox]
{
    background-color: #DDD;
    border-radius: 2px;
    appearance: none;
    -webkit-appearance: none;
    -moz-appearance: none;
    width: 17px;
    height: 17px;
    cursor: pointer;
    position: relative;
    top: 5px;
}
```

```
}  
  
INPUT[type=checkbox]:checked  
{  
  background-color: #409fd6;  
  background: #409fd6  
  url("data:image/gif;base64,R0lGODlhCwAKAIABAP////3cnSH5BAEKAAEALAAAAAALAAoAAAIUj  
3px 3px no-repeat;  
}
```

```
<form>  
  <label><input type="checkbox"> I Agree To Terms & Conditions</label>  
</form>
```



Run code snippet

[Expand snippet](#)

Share Improve this answer

edited Nov 24, 2020 at 4:15

answered Jun 2, 2019 at 2:13

Follow



Volomike

24.8k ● 22 ● 122 ● 216

- 5 Great idea, Volomike. Some improvements: 1) set width and height to 1em and top to 0.2em to make it scalable with the font size. 2) set background to center/cover to scale with the size. Also, the `background-color` in your code is redundant. 3) remove the `cursor: pointer` - keep the default behavior and let user set it in their stylesheet. – Jan Turoň Aug 29, 2021 at 8:10
- 1 You can also just use: `all: unset`, for the checkbox, which has almost identical support out of the box, without requiring the browser-specific flags, and the caveats that come along with `appearance`. :) – Jack\_Hu Feb 13, 2022 at 12:40



37



## Simple to implement and easily customizable solution

After a lot of search and testing I got this solution which is simple to implement and easier to customize. **In this solution:**

1. You don't need external libraries and files
2. You don't need to add extra HTML in your page
3. You don't need to change checkbox names and id

Simple put the flowing CSS at the top of your page and all checkboxes style will change like this:

Add User



Agent Payments



Update Agent Credit



```
input[type=checkbox] {
  transform: scale(1.5);
}

input[type=checkbox] {
  width: 30px;
  height: 30px;
  margin-right: 8px;
  cursor: pointer;
  font-size: 17px;
  visibility: hidden;
}

input[type=checkbox]:after,
input[type=checkbox]::after {
  content: " ";
  background-color: #fff;
  display: inline-block;
  margin-left: 10px;
  padding-bottom: 5px;
  color: #00BFF0;
  width: 22px;
  height: 25px;
  visibility: visible;
  border: 1px solid #00BFF0;
  padding-left: 3px;
  border-radius: 5px;
}

input[type=checkbox]:checked:after,
input[type=checkbox]:checked::after {
  content: "\2714";
  padding: -5px;
  font-weight: bold;
}
```

```
<input type="checkbox" id="checkbox1" />
<label for="checkbox1">Checkbox</label>
```

▶ Run code snippet

🔗 [Expand snippet](#)

Share Improve this answer

edited Mar 25, 2022 at 10:29

answered Jan 4, 2017 at 21:45

Follow



X3R0

6,274 ● 2 ● 16 ● 44



Mohamed Nor

613 ● 1 ● 10 ● 12

- 4 Nice solution. But I dont see it working on Firefox, IE or Edge (there is no checkbox) – Jono Mar 6, 2019 at 15:00

@Jono I've updated the answer, please try it now. – X3R0 Mar 25, 2022 at 10:29

- 2 @DeanVanGreunen still no checkbox on Firefox (Mac, 97.0.2) - I am not able to check IE or Edge – Jono Mar 27, 2022 at 12:31

I see a huge checkbox on Edge ([i.sstatic.net/kOhLI.png](https://i.sstatic.net/kOhLI.png)), but yes it is blank in Firefox (no blue checkbox shown or even clickable). cc @Jono IE is not supported by Stack Overflow and Stack Snippets won't run on that browser. Nor will most other code sandbox sites like JSFiddle or CodePen, to my knowledge... – TylerH Apr 27, 2022 at 21:05

You can style checkboxes with a little trickery using the `label` element an example is below:

35

```
.checkbox > input[type=checkbox] {
  visibility: hidden;
}

.checkbox {
  position: relative;
  display: block;
  width: 80px;
  height: 26px;
  margin: 0 auto;
  background: #FFF;
  border: 1px solid #2E2E2E;
  border-radius: 2px;
  -webkit-border-radius: 2px;
  -moz-border-radius: 2px;
}

.checkbox:after {
  position: absolute;
  display: inline;
  right: 10px;
  content: 'no';
  color: #E53935;
  font: 12px/26px Arial, sans-serif;
  font-weight: bold;
  text-transform: capitalize;
  z-index: 0;
}

.checkbox:before {
  position: absolute;
  display: inline;
  left: 10px;
  content: 'yes';
  color: #43A047;
}
```

```
font: 12px/26px Arial, sans-serif;
font-weight: bold;
text-transform: capitalize;
z-index: 0;
}

.checkbox label {
position: absolute;
display: block;
top: 3px;
left: 3px;
width: 34px;
height: 20px;
background: #2E2E2E;
cursor: pointer;
transition: all 0.5s linear;
-webkit-transition: all 0.5s linear;
-moz-transition: all 0.5s linear;
border-radius: 2px;
-webkit-border-radius: 2px;
-moz-border-radius: 2px;
z-index: 1;
}

.checkbox input[type=checkbox]:checked + label {
left: 43px;
}
```

```
<div class="checkbox">
  <input id="checkbox1" type="checkbox" value="1" />
  <label for="checkbox1"></label>
</div>
```



Run code snippet

[Expand snippet](#)

And a [FIDDLE](#) for the above code. Note that some CSS doesn't work in older versions of browsers, but I'm sure there are some fancy JavaScript examples out there!

Share Improve this answer

Follow

edited Apr 26, 2018 at 13:58



Vadim Ovchinnikov

14k ● 7 ● 65 ● 94

answered Aug 17, 2013 at 21:22



TURTLE

3,847 ● 4 ● 53 ● 53

How to use it where the checkbox has its own label? – [Metaphor](#) Sep 10, 2020 at 20:17

@Metaphor Just add a second label. You can apply multiple labels to the same element (e.g. they can use the same `for` attribute value). – [TylerH](#) Nov 5, 2021 at 13:55



You can avoid adding extra markup. This works everywhere except IE via setting CSS appearance :



```
input[type="checkbox"] {
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;

  /* Styling checkbox */
  width: 16px;
  height: 16px;
  background-color: red;
}

input[type="checkbox"]:checked {
  background-color: green;
}
```

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

[Run code snippet](#)[Expand snippet](#)

Share Improve this answer

edited Jun 16, 2022 at 11:07

answered Jun 30, 2017 at 15:40

Follow



Vadim Ovchinnikov

14k ● 7 ● 65 ● 94

- 4 I like this approach - it is a bit hacky as you shouldn't ever really use appearance in css but it is much cleaner than some of the other answers using before and after. And for something like this I feel JavaScript is complete overkill. – [Sprose](#) Jul 18, 2017 at 9:41



25



Recently I found a quite interesting solution to the problem.

You could use `appearance: none;` to turn off the checkbox's default style and then write your own over it like described [here \(Example 4\)](#).

```
input[type=checkbox] {
  width: 23px;
  height: 23px;
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;
  margin-right: 10px;
  background-color: #878787;
  outline: 0;
  border: 0;
  display: inline-block;
  -webkit-box-shadow: none !important;
  -moz-box-shadow: none !important;
  box-shadow: none !important;
}

input[type=checkbox]:focus {
```



```
outline: none;
border: none !important;
-webkit-box-shadow: none !important;
-moz-box-shadow: none !important;
box-shadow: none !important;
}

input[type=checkbox]:checked {
background-color: green;
text-align: center;
line-height: 15px;
}
```

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



Run code snippet

[Expand snippet](#)

Unfortunately browser support is quite bad for the `appearance` option. From my personal testing I only got Opera and Chrome working correctly. But this would be the way to go to keep it simple when better support comes or you only want to use Chrome/Opera.

[Example JSFiddle](#)

["Can I use?" link](#)

Share Improve this answer

Follow

edited Oct 1, 2021 at 14:05



TylerH

21.2k ● 76 ● 79 ● 110

answered Dec 3, 2015 at 8:24



Joost

761 ● 7 ● 18

Indeed `appearance` is just what we need for this; just bear in mind it ["is non-standard and is not on a standards track"](#). – sam Feb 12, 2016 at 3:21

I've added a little more styling to make it more closely match chrome tick box. So its easily interchangeable: - [jsfiddle.net/inspiraller/g8mo1nh3/1](#) – Inspiraller Oct 5, 2021 at 11:53

One of the downsides here is that you lose the "check" part of "checkbox". It's now just a box that changes color, which is far less clear from a UX/accessibility perspective as to what it is, and what's happening when you click it. – TylerH Nov 5, 2021 at 13:52



22



I prefer to use icon fonts (such as FontAwesome) since it's easy to modify their colours with CSS, and they scale really well on high pixel-density devices. So here's another pure CSS variant, using similar techniques to those above.

(Below is a static image so you can visualize the result; see [the JSFiddle](#) for an interactive version.)





# Cheese



As with other solutions, it uses the `label` element. An adjacent `span` holds our checkbox character.

```
span.bigcheck-target {
  font-family: FontAwesome; /* Use an icon font for the checkbox */
}

input[type='checkbox'].bigcheck {
  position: relative;
  left: -999em; /* Hide the real checkbox */
}

input[type='checkbox'].bigcheck + span.bigcheck-target:after {
  content: "\f096"; /* In fontawesome, is an open square (fa-square-o) */
}

input[type='checkbox'].bigcheck:checked + span.bigcheck-target:after {
  content: "\f046"; /* fontawesome checked box (fa-check-square-o) */
}

/* ==== Optional - colors and padding to make it look nice === */
body {
  background-color: #2C3E50;
  color: #D35400;
  font-family: sans-serif;
  font-weight: 500;
  font-size: 4em; /* Set this to whatever size you want */
}

span.bigcheck {
  display: block;
  padding: 0.5em;
}
```

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" />
```

```
<span class="bigcheck">
  <label class="bigcheck">
    Cheese
    <input type="checkbox" class="bigcheck" name="cheese" value="yes" />
    <span class="bigcheck-target"></span>
  </label>
</span>
```



Run code snippet

[Expand snippet](#)

Here's the [JSFiddle](#) for it.

Share Improve this answer

Follow

edited Jul 17, 2019 at 18:02



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Jul 2, 2014 at 7:02



cobberboy

6,175 ● 2 ● 30 ● 23

1 ....what you described is the same as previous answers from Bhushan wagh, Jake, Jonathan Hodgson and Blake ;) – [SW4](#) Jan 26, 2015 at 8:53

8 @SW4 I've made reference to that fact, except this answer uses icon fonts (which none of the previous answers had). The first paragraph makes that pretty clear. – [cobberboy](#) Jan 26, 2015 at 17:12

You need to replace `font-family: FontAwesome;` with `font-family: 'Font Awesome\ 5 Free';`, and update unicode content if you want to make it work in new version. – [SuN](#) Feb 6, 2019 at 11:44 ✎



## My solution

17



```
input[type="checkbox"] {
  cursor: pointer;
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;
  outline: 0;
  background: lightgray;
  height: 16px;
  width: 16px;
  border: 1px solid white;
}

input[type="checkbox"]:checked {
  background: #2aa1c0;
}

input[type="checkbox"]:hover {
  filter: brightness(90%);
}

input[type="checkbox"]:disabled {
  background: #e6e6e6;
  opacity: 0.6;
  pointer-events: none;
}

input[type="checkbox"]:after {
  content: '';
  position: relative;
  left: 40%;
  top: 20%;
  width: 15%;
  height: 40%;
  border: solid #fff;
  border-width: 0 2px 2px 0;
  transform: rotate(45deg);
}
```

```
display: none;
}

input[type="checkbox"]:checked:after {
  display: block;
}

input[type="checkbox"]:disabled:after {
  border-color: #7b7b7b;
}
```

```
<input type="checkbox"><br>
<input type="checkbox" checked><br>
<input type="checkbox" disabled><br>
<input type="checkbox" disabled checked><br>
```

▶ Run code snippet

🔗 [Expand snippet](#)

Share Improve this answer

edited Apr 19, 2018 at 18:54

answered Apr 19, 2018 at 18:35

Follow



[agirault](#)

2,918 ● 21 ● 24

If someone could advice why my text is not aligned after, I'd love to know! CSS beginner here.  
– [agirault](#) Apr 19, 2018 at 18:56

Checked the entire thread and appearance seems to be key as well as after. Also, if someone need to have a larger checkbox, then this solution will work just fine. Just play around with top, left, width, height and border-width. Works in modern browsers according to caniuse.com. I'm writing this comment in May 2021 for my future self. :) – [PussInBoots](#) May 5, 2021 at 16:27 ✎



You can simply use `appearance: none` on modern browsers, so that there is no default styling and all your styles are applied properly:

14



```
input[type=checkbox] {
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;
  display: inline-block;
  width: 2em;
  height: 2em;
  border: 1px solid gray;
  outline: none;
  vertical-align: middle;
}

input[type=checkbox]:checked {
  background-color: blue;
}
```



collimarco

35.3k ● 37 ● 113 ● 155

1 This copies the solution from Joost's answer from 2015. – TylerH Sep 13, 2021 at 14:38

@TylerH Do you know what *copy* means? The fact that we both used `appearance` doesn't mean that the answer is the same. My code is definitely smaller and cleaner and I also add that the solution works on modern browsers. – collimarco Sep 14, 2021 at 11:03

I do, in fact, know what copy means. [Joost's answer](#) suggests using `appearance: none` as a solution. This answer also suggests using `appearance: none` as a solution. Since it suggests the same solution, it's a copy. It does not have to be a *literal character-for-character facsimile* to be a copy. Next time, please read all existing answers before posting your own, to avoid simply repeating information. – TylerH Sep 14, 2021 at 13:10

Here is a simple CSS solution without any jQuery or JavaScript code.

I am using FontAwseome icons but you can use any image

```
input[type=checkbox] {
  display: inline-block;
  font-family: FontAwesome;
  font-style: normal;
  font-weight: normal;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  visibility: hidden;
  font-size: 14px;
}

input[type=checkbox]:before {
  content: @fa-var-square-o;
  visibility: visible;
  /*font-size: 12px;*/
}

input[type=checkbox]:checked:before {
  content: @fa-var-check-square-o;
}
```

Share Improve this answer

Follow

edited Jul 17, 2019 at 18:03



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 1, 2014 at 11:38



aWebDeveloper

38.3k ● 41 ● 175 ● 246

1 Unlike most of the other answers, this one doesn't require creation of additional `label` or `span` elements. It just works! – KurtPreston Feb 9, 2017 at 15:22

Could you show an example or a demo, or at least the HTML part, with which this CSS code is working? @aWebDeveloper – iorgu Apr 2, 2017 at 1:38

@aWebDeveloper making inane edits to bump an answer or question to the home page is against the rules. – TylerH Jul 12, 2017 at 16:06

@aWebDeveloper You edited an old answer on an inactive question, which bumps it to the front page and provides a link straight to your answer. This is fine if you have pertinent info to add or change in an answer. All you did was embed one of your sentences in a quote block, which isn't useful to anyone. – TylerH Jul 25, 2017 at 20:36

should work @VadimOvchinnikov... nothing here is chrome specific – aWebDeveloper Jul 26, 2017 at 15:05

From my googling, this is the easiest way for checkbox styling. Just add `:after` and `:checked:after` CSS based on your design.

```
body{
  background: #DDD;
}
span{
  margin-left: 30px;
}
input[type=checkbox] {
  cursor: pointer;
  font-size: 17px;
  visibility: hidden;
  position: absolute;
  top: 0;
  left: 0;
  transform: scale(1.5);
}

input[type=checkbox]:after {
  content: " ";
  background-color: #fff;
  display: inline-block;
  color: #00BFF0;
  width: 14px;
  height: 19px;
  visibility: visible;
  border: 1px solid #FFF;
  padding: 0 3px;
  margin: 2px 0;
  border-radius: 8px;
  box-shadow: 0 0 15px 0 rgba(0,0,0,0.08), 0 0 2px 0 rgba(0,0,0,0.16);
}

input[type=checkbox]:checked:after {
  content: "\2714";
  display: unset;
  font-weight: bold;
}
```

```
<input type="checkbox"> <span>Select Text</span>
```

[Run code snippet](#)[Expand snippet](#)[Share](#) [Improve this answer](#)[Follow](#)

edited Jul 17, 2019 at 22:44

[Peter Mortensen](#)

31.6k ● 22 ● 109 ● 133

answered May 15, 2018 at 6:20

[Mahfuzur Rahman](#)

1,535 ● 20 ● 24



7



Modify the checkbox style with plain CSS. This does not require any JavaScript or HTML manipulation:

```
.form input[type="checkbox"]:before {
  display: inline-block;
  font: normal normal normal 14px/1 FontAwesome;
  font-size: inherit;
  text-rendering: auto;
  -webkit-font-smoothing: antialiased;
  content: "\f096";
  opacity: 1 !important;
  margin-top: -25px;
  appearance: none;
  background: #fff;
}

.form input[type="checkbox"]:checked:before {
  content: "\f046";
}

.form input[type="checkbox"] {
  font-size: 22px;
  appearance: none;
  -webkit-appearance: none;
  -moz-appearance: none;
}
```

```
<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet" />
```

```
<form class="form">
  <input type="checkbox" />
</form>
```

[Run code snippet](#)[Expand snippet](#)[Share](#) [Improve this answer](#)[Follow](#)

edited May 2, 2022 at 17:46

[maxshuty](#)


10.6k ● 13 ● 69 ● 85

answered Dec 16, 2016 at 10:13

[BEJGAM SHIVA PRASAD](#)

2,241 ● 1 ● 20 ● 28

---

@VadimOvchinnikov We have one solution which requires to following some html structure..[stackoverflow.com/questions/23305780/...](https://stackoverflow.com/questions/23305780/) – BEJGAM SHIVA PRASAD Aug 17, 2017 at 15:24 

---



Yikes! All these workarounds have led me to the conclusion that the HTML checkbox kind of sucks if you want to style it.

6



As a forewarning, this isn't a CSS implementation. I just thought I'd share the workaround I came up with in case anyone else might find it useful.



I used the HTML5 `canvas` element.



The upside to this is that you don't have to use external images and can probably save some bandwidth.

The downside is that if a browser for some reason can't render it correctly, then there's no fallback. Though whether this remains an issue in 2017 is debatable.

## Update

I found the old code quite ugly, so I decided to give it a rewrite.

```
Object.prototype.create = function(args){
    var retobj = Object.create(this);

    retobj.constructor(args || null);

    return retobj;
}

var Checkbox = Object.seal({
    width: 0,
    height: 0,
    state: 0,
    document: null,
    parent: null,
    canvas: null,
    ctx: null,

    /*
     * args:
     * name      default      desc.
     *
     * width      15           width
     * height     15           height
     * document   window.document  explicit document reference
     * target     this.document.body  target element to insert checkbox into
     */
    constructor: function(args){
        if(args === null)
            args = {};
    }
});
```



```

        this.width = args.width || 15;
        this.height = args.height || 15;
        this.document = args.document || window.document;
        this.parent = args.target || this.document.body;
        this.canvas = this.document.createElement("canvas");
        this.ctx = this.canvas.getContext('2d');

        this.canvas.width = this.width;
        this.canvas.height = this.height;
        this.canvas.addEventListener("click", this.ev_click(this), false);
        this.parent.appendChild(this.canvas);
        this.draw();
    },

    ev_click: function(self){
        return function(unused){
            self.state = !self.state;
            self.draw();
        }
    },

    draw_rect: function(color, offset){
        this.ctx.fillStyle = color;
        this.ctx.fillRect(offset, offset,
            this.width - offset * 2, this.height - offset * 2);
    },

    draw: function(){
        this.draw_rect("#CCCCCC", 0);
        this.draw_rect("#FFFFFF", 1);

        if(this.is_checked())
            this.draw_rect("#000000", 2);
    },

    is_checked: function(){
        return !!this.state;
    }
});

```

Here's a [working demo](#).

The new version uses prototypes and differential inheritance to create an efficient system for creating checkboxes. To create a checkbox:

```
var my_checkbox = Checkbox.create();
```

This will immediately add the checkbox to the DOM and hook up the events. To query whether a checkbox is checked:

```
my_checkbox.is_checked(); // True if checked, else false
```

Also important to note is that I got rid of the loop.

## Update 2

Something I neglected to mention in the last update is that using the canvas has more advantages than just making a checkbox that looks however you want it to look. You could also create *multi-state* checkboxes, if you wanted to.

```
Object.prototype.create = function(args){
    var retobj = Object.create(this);

    retobj.constructor(args || null);

    return retobj;
}

Object.prototype.extend = function(newobj){
    var oldobj = Object.create(this);

    for(prop in newobj)
        oldobj[prop] = newobj[prop];

    return Object.seal(oldobj);
}

var Checkbox = Object.seal({
    width: 0,
    height: 0,
    state: 0,
    document: null,
    parent: null,
    canvas: null,
    ctx: null,

    /*
     * args:
     * name      default      desc.
     *
     * width      15           width
     * height     15           height
     * document   window.document  explicit document reference
     * target     this.document.body  target element to insert checkbox into
     */
    constructor: function(args){
        if(args === null)
            args = {};

        this.width = args.width || 15;
        this.height = args.height || 15;
        this.document = args.document || window.document;
        this.parent = args.target || this.document.body;
        this.canvas = this.document.createElement("canvas");
        this.ctx = this.canvas.getContext('2d');

        this.canvas.width = this.width;
        this.canvas.height = this.height;
        this.canvas.addEventListener("click", this.ev_click(this), false);
        this.parent.appendChild(this.canvas);
        this.draw();
    },

    ev_click: function(self){
```

```

        return function(unused){
            self.state = !self.state;
            self.draw();
        }
    },

    draw_rect: function(color, offsetx, offsety){
        this.ctx.fillStyle = color;
        this.ctx.fillRect(offsetx, offsety,
            this.width - offsetx * 2, this.height - offsety * 2);
    },

    draw: function(){
        this.draw_rect("#CCCCCC", 0, 0);
        this.draw_rect("#FFFFFF", 1, 1);
        this.draw_state();
    },

    draw_state: function(){
        if(this.is_checked())
            this.draw_rect("#000000", 2, 2);
    },

    is_checked: function(){
        return this.state == 1;
    }
});

var Checkbox3 = Checkbox.extend({
    ev_click: function(self){
        return function(unused){
            self.state = (self.state + 1) % 3;
            self.draw();
        }
    },

    draw_state: function(){
        if(this.is_checked())
            this.draw_rect("#000000", 2, 2);

        if(this.is_partial())
            this.draw_rect("#000000", 2, (this.height - 2) / 2);
    },

    is_partial: function(){
        return this.state == 2;
    }
});

```

I modified slightly the `Checkbox` used in the last snippet so that it is more generic, making it possible to "extend" it with a checkbox that has 3 states. Here's [a demo](#). As you can see, it already has more functionality than the built-in checkbox.

Something to consider when you're choosing between JavaScript and CSS.

## Old, poorly-designed code

[Working Demo](#)

First, set up a canvas

```
var canvas = document.createElement('canvas'),
    ctx = canvas.getContext('2d'),
    checked = 0; // The state of the checkbox
canvas.width = canvas.height = 15; // Set the width and height of the canvas
document.body.appendChild(canvas);
document.body.appendChild(document.createTextNode(' Toggleable Option'));
```

Next, devise a way to have the canvas update itself.

```
(function loop(){
  // Draws a border
  ctx.fillStyle = '#ccc';
  ctx.fillRect(0,0,15,15);
  ctx.fillStyle = '#fff';
  ctx.fillRect(1, 1, 13, 13);
  // Fills in canvas if checked
  if(checked){
    ctx.fillStyle = '#000';
    ctx.fillRect(2, 2, 11, 11);
  }
  setTimeout(loop, 1000/10); // Refresh 10 times per second
})();
```

The last part is to make it interactive. Luckily, it's pretty simple:

```
canvas.onclick = function(){
  checked = !checked;
}
```

This is where you might have problems in IE, due to their weird event handling model in JavaScript.

I hope this helps someone; it definitely suited my needs.

Share Improve this answer

edited Jul 18, 2019 at 2:26

answered Sep 24, 2013 at 23:16

Follow



**Braden Best**

8,998 ● 4 ● 33 ● 46

Note that the Canvas implementation, as an emulation of checkboxes, allows for more functionality than built-in checkboxes. For example, fancy stuff like "multi-state" checkboxes (i.e. unchecked (e.g. unchecked -> checked -> "kinda" checked -> unchecked); the states could represent "explicit false", "explicit true", "use default", for example). I'm considering adding an example to the answer. – [Braden Best](#) Feb 20, 2017 at 6:37

1 Terrible idea in my opinion. – [user147215](#) Feb 15, 2018 at 15:02

@Neil please elaborate – [Braden Best](#) Feb 15, 2018 at 17:56

1 Why reinvent the wheel? – [user147215](#) Feb 21, 2018 at 19:32

@Neil why *not* reinvent the wheel? NIH isn't always a bad thing, especially when it affords advantages that would otherwise be impossible or stupidly complicated to implement with the existing tech. So it's a question of whether you're willing to deal with the built-in check boxes. And if I am wanting to theme them to blend with the visual language of my website or app, I want full control. So until someone makes a standalone UI library that's lightweight and easy to use, I personally prefer my wheels to be invented here. Someone else might not, but that doesn't invalidate my answer. – [Braden Best](#) Feb 22, 2018 at 5:08



## SCSS / SASS Implementation

5

### A more modern approach



For those using SCSS (or easily converted to SASS), the following will be helpful. Effectively, make an element next to the checkbox, which is the one that you will style.



When the checkbox is clicked, the CSS restyles the sister element (to your new, checked style). Code is below:



```
label.checkbox {
  input[type="checkbox"] {
    visibility: hidden;
    display: block;
    height: 0;
    width: 0;
    position: absolute;
    overflow: hidden;

    &:checked + span {
      background: $accent;
    }
  }

  span {
    cursor: pointer;
    height: 15px;
    width: 15px;
    border: 1px solid $accent;
    border-radius: 2px;
    display: inline-block;
    transition: all 0.2s $interpol;
  }
}
```

```
<label class="checkbox">
  <input type="checkbox" />
  <span></span>
  Label text
</label>
```



Run code snippet

[Expand snippet](#)



A simple and lightweight template as well:

```
input[type=checkbox] {
  cursor: pointer;
}

input[type=checkbox]:checked:before {
  content: "\2713";
  background: #fffed5;
  text-shadow: 1px 1px 1px rgba(0, 0, 0, .2);
  font-size: 20px;
  text-align: center;
  line-height: 8px;
  display: inline-block;
  width: 13px;
  height: 15px;
  color: #00904f;
  border: 1px solid #cdcdcd;
  border-radius: 4px;
  margin: -3px -3px;
  text-indent: 1px;
}

input[type=checkbox]:before {
  content: "\202A";
  background: #ffffff;
  text-shadow: 1px 1px 1px rgba(0, 0, 0, .2);
  font-size: 20px;
  text-align: center;
  line-height: 8px;
  display: inline-block;
  width: 13px;
  height: 15px;
  color: #00904f;
  border: 1px solid #cdcdcd;
  border-radius: 4px;
  margin: -3px -3px;
  text-indent: 1px;
}
```

```
<input type="checkbox" checked="checked">checked1<br>
<input type="checkbox">unchecked2<br>
<input type="checkbox" checked="checked" id="id1">
<label for="id1">checked2+label</label><br>
<label for="id2">unchecked2+label+rtl</label>
<input type="checkbox" id="id2">
<br>
```

[Run code snippet](#)[Expand snippet](#)

<https://jsfiddle.net/rvgccn5b/>

Share Improve this answer

Follow

edited Apr 26, 2018 at 14:31



Vadim Ovchinnikov

14k ● 7 ● 65 ● 94

answered Aug 16, 2017 at 8:27



javad shariaty

989 ● 10 ● 15

- 1 Pseudoelements for `input` s are supported only in Chrome, your code doesn't work the same in every browser. – Vadim Ovchinnikov Apr 26, 2018 at 14:33



4

HTML



```
<input type="checkbox" id="first" />
<label for="first">&nbsp;</label>
```



CSS

```
checkbox {
  display: none;
}

checkbox + label {
  /* Style for checkbox normal */
  width: 16px;
  height: 16px;
}

checkbox:checked + label,
label.checked {
  /* Style for checkbox checked */
}
```

The `checkbox`, even though it is hidden, will still be accessible, and its value will be sent when a form is submitted. For old browsers you might have to change the class of the `label` to checked using JavaScript because I don't think old versions of Internet Explorer understand `::checked` on the `checkbox`.

Share Improve this answer

Follow

edited Jul 17, 2019 at 17:48



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Nov 24, 2012 at 19:28



Jonathan Hodgson

89 ● 1 ● 1

- 4 The difference between your answer and mine is what exactly? – Blake Pettersson Dec 5, 2012 at 17:50

@BlakePettersson yours is correct, ::checked is just wrong (I even tried just in case it's also allowed) ;) – [esp](#) Feb 24, 2013 at 0:31

- 11 This is a bad answer. (a) ::checked is wrong—it should be :checked . (b) checkbox is wrong—it should be [type=checkbox] – [Chris Morgan](#) Feb 4, 2014 at 5:12

You should not hide checkbox with display: none or visibility: none , as this breaks accessibility – [James Jenkinson](#) Oct 25, 2022 at 10:30



No JavaScript or jQuery required.

4

Change your checkbox style simple way.



```
input[type="checkbox"] {
  display: none;
  border: none !important;
  box-shadow: none !important;
}

input[type="checkbox"] + label span {
  background: url("https://i.ibb.co/mG7QRYw/62a9b364a056b98f7e02705a-
checkboxunselected.png");
  width: 24px;
  height: 24px;
  display: inline-block;
  vertical-align: middle;
  background-size: 100%;
}

input[type="checkbox"]:checked + label span {
  background: url(https://svgur.com/i/upi.svg);
  width: 24px;
  height: 24px;
  vertical-align: middle;
  background-size: 100%;
}
```

```
<input type="checkbox" id="option" />
<label for="option"> <span></span> Click me </label>
```



Run code snippet

[Expand snippet](#)

[Here is a JSFiddle link](#)

Share Improve this answer

Follow

edited Jul 1, 2023 at 0:09



[Tiago Rangel](#)

1,318 ● 19 ● 34


answered Apr 6, 2016 at 10:46



[Cherish](#)

312 ● 1 ● 7



- 
- 3 The checkbox isn't showing up, because it relies on external links to image files [img.h.us/uncheck.png](http://img.h.us/uncheck.png) and [img.h.us/check\\_2.png](http://img.h.us/check_2.png) which are no longer available online. – [jkdev](#) Jan 31, 2020 at 4:05 
- 

@jkdev I edited it so now the links are online. still waiting for approval – [Tiago Rangel](#) Jun 30, 2023 at 15:57

---



3



No, you still can't style the checkbox itself, but I (finally) figured out how to style an illusion while **keeping the functionality** of clicking a checkbox. It means that you can toggle it even if the cursor isn't perfectly still without risking selecting text or triggering drag-and-drop!

This solution probably also fits radio buttons.

The following works in Internet Explorer 9, Firefox 30.0 and Chrome 40.0.2214.91 and is just a basic example. You can still use it in combination with background images and pseudo-elements.

<http://jsfiddle.net/o0xo13yL/1/>

```
label {
  display: inline-block;
  position: relative; /* Needed for checkbox absolute positioning */
  background-color: #eee;
  padding: .5rem;
  border: 1px solid #000;
  border-radius: .375rem;
  font-family: "Courier New";
  font-size: 1rem;
  line-height: 1rem;
}

label > input[type="checkbox"] {
  display: block;
  position: absolute; /* Remove it from the flow */
  width: 100%;
  height: 100%;
  margin: -.5rem; /* Negative the padding of label to cover the "button" */
  cursor: pointer;
  opacity: 0; /* Make it transparent */
  z-index: 666; /* Place it on top of everything else */
}

label > input[type="checkbox"] + span {
  display: inline-block;
  width: 1rem;
  height: 1rem;
  border: 1px solid #000;
  margin-right: .5rem;
}

label > input[type="checkbox"]:checked + span {
  background-color: #666;
}
```

```
<label>
  <input type="checkbox" />
  <span>&nbsp;</span>Label text
</label>
```

Share Improve this answer

Follow

edited Jul 17, 2019 at 22:23



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Jan 24, 2015 at 11:59



LGT

5,034 ● 1 ● 24 ● 22

Here's a modern version with a little animation, and simple styling you can customize:

3

```
.checkbox {
  position: relative;
  width: 20px;
  height: 20px;
  -webkit-appearance: none;
  -moz-appearance: none;
  -o-appearance: none;
  appearance: none;
  background: transparent;
  border: 2px solid #7C7A7D;
  border-radius: 5px;
  margin: 0;
  outline: none;
  transition: 0.5s ease;
  opacity: 0.8;
  cursor: pointer;
}

.checkbox:checked {
  border-color: #7C7A7D;
  background-color: #7C7A7D;
}

.checkbox:checked:before {
  position: absolute;
  left: 2px;
  top: -4px;
  display: block;
  content: '\2713';
  text-align: center;
  color: #FFF;
  font-family: Arial;
  font-size: 14px;
  font-weight: 800;
}

.checkbox:hover {
  opacity: 1.0;
  transform: scale(1.05);
}
```



3



**Custom checkbox with CSS** (WebKit browser solution only Chrome, Safari, Mobile browsers)

```
<input type="checkbox" id="cardAcceptance" name="cardAcceptance" value="Yes">
<label for="cardAcceptance" class="bold"> Save Card for Future Use</label>
```

CSS:

```
/* The checkbox-cu */

.checkbox-cu {
  display: block;
  position: relative;
  padding-left: 35px;
  margin-bottom: 0;
  cursor: pointer;
  font-size: 16px;
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}

/* Hide the browser's default checkbox-cu */

.checkbox-cu input {
  position: absolute;
  opacity: 0;
  cursor: pointer;
  height: 0;
  width: 0;
}

/* Create a custom checkbox-cu */

.checkmark {
  position: absolute;
  top: 4px;
  left: 0;
  height: 20px;
  width: 20px;
  background-color: #eee;
  border: 1px solid #999;
  border-radius: 0;
  box-shadow: none;
}

/* On mouse-over, add a grey background color */

.checkbox-cu:hover input~.checkmark {
```

```

    background-color: #ccc;
}

/* When the checkbox-cu is checked, add a blue background */

.checkbox-cu input:checked~.checkmark {
    background-color: transparent;
}

/* Create the checkmark/indicator (hidden when not checked) */

.checkmark:after {
    content: "";
    position: absolute;
    display: none;
}

/* Show the checkmark when checked */

.checkbox-cu input:checked~.checkmark:after {
    display: block;
}

/* Style the checkmark/indicator */

.checkbox-cu .checkmark::after {
    left: 7px;
    top: 3px;
    width: 6px;
    height: 9px;
    border: solid #28a745;
    border-width: 0 2px 2px 0;
    -webkit-transform: rotate(45deg);
    -ms-transform: rotate(45deg);
    transform: rotate(45deg);
    z-index: 100;
}

```

Share Improve this answer

edited Sep 13, 2021 at 14:40

answered Sep 20, 2019 at 7:04

Follow



TylerH

21.2k ● 76 ● 79 ● 110



Sonu Nagar

134 ● 5

By using [Materialize](#) with a custom stylesheet, you can achieve something like this:

### CSS code

```

.custom_checkbox[type="checkbox"]:checked + span:not(.lever)::before {
    border: 2px solid transparent;
    border-bottom: 2px solid #ffd600;
    border-right: 2px solid #ffd600;
    background: transparent;
}

```

## HTML code

```
<label>
  <input type="checkbox" class="custom_checkbox" />
  <span>Text</span>
</label>
```

## Demo

[JSFiddle demo](#)

Share Improve this answer

Follow

edited Sep 13, 2021 at 14:41



TylerH

21.2k ● 76 ● 79 ● 110

answered May 13, 2021 at 21:01



Riccardo Volpe

1,583 ● 1 ● 16 ● 33



This helped me to change style (color) for checkbox

3



```
input[type=checkbox] {
  accent-color: red;
}
```



We can also use the same for radio buttons.



Share Improve this answer Follow

answered Jan 10, 2022 at 6:54



p.durga shankar

1,207 ● 10 ● 21



This is simplest way and you can choose which checkboxes to give this style.

2



CSS:

```
.check-box input {
  display: none;
}

.check-box span:before {
  content: ' ';
  width: 20px;
  height: 20px;
  display: inline-block;
  background: url("unchecked.png");
}

.check-box input:checked + span:before {
  background: url("checked.png");
}
```



HTML:

```
<label class="check-box">
  <input type="checkbox">
  <span>Check box Text</span>
</label>
```

Share Improve this answer

edited Apr 26, 2018 at 14:58

answered Jul 21, 2015 at 13:40

Follow



Vadim Ovchinnikov

14k ● 7 ● 65 ● 94



Ali Gonabadi

944 ● 1 ● 10 ● 28



Here is a CSS/HTML-only version, no jQuery or JavaScript needed at all, Simple and clean HTML and really simple and short CSS.

2

Here is the JSFiddle



<http://jsfiddle.net/v71kn3pr/>



Here is the HTML:



```
<div id="myContainer">
  <input type="checkbox" name="myCheckbox" id="myCheckbox_01_item"
  value="red" />
  <label for="myCheckbox_01_item" class="box"></label>
  <label for="myCheckbox_01_item" class="text">I accept the Terms of Use.
</label>
</div>
```

Here is the CSS

```
#myContainer {
  outline: black dashed 1px;
  width: 200px;
}
#myContainer input[type="checkbox"][name="myCheckbox"] {
  display: none;
}
#myContainer input[type="checkbox"][name="myCheckbox"]:not(:checked) +
label.box {
  display: inline-block;
  width: 25px;
  height: 25px;
  border: black solid 1px;
  background: #FFF ;
  margin: 5px 5px;
}
#myContainer input[type="checkbox"][name="myCheckbox"]:checked + label.box {
  display: inline-block;
  width: 25px;
  height: 25px;
  border: black solid 1px;
  background: #F00;
  margin: 5px 5px;
}
#myContainer input[type="checkbox"][name="myCheckbox"] + label + label.text {
```

```
font: normal 12px arial;  
display: inline-block;  
line-height: 27px;  
vertical-align: top;  
margin: 5px 0px;  
}
```

This can be adapted to be able to have individual radio or checkboxes, groups of checkboxes and groups of radio buttons as well.

This html/css, will allow you to also capture click on the label, so the checkbox will be checked and unchecked even if you click just on the label.

This type of checkbox/radio button works perfectly with any form, no problem at all. Have been tested using PHP, ASP.NET (.aspx), [JavaServer Faces](#), and [ColdFusion](#) too.

Share Improve this answer

Follow

edited Jul 17, 2019 at 18:08



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 4, 2014 at 3:28



Allan

261 ● 1 ● 7

1

2

Next



**Highly active question.** Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.