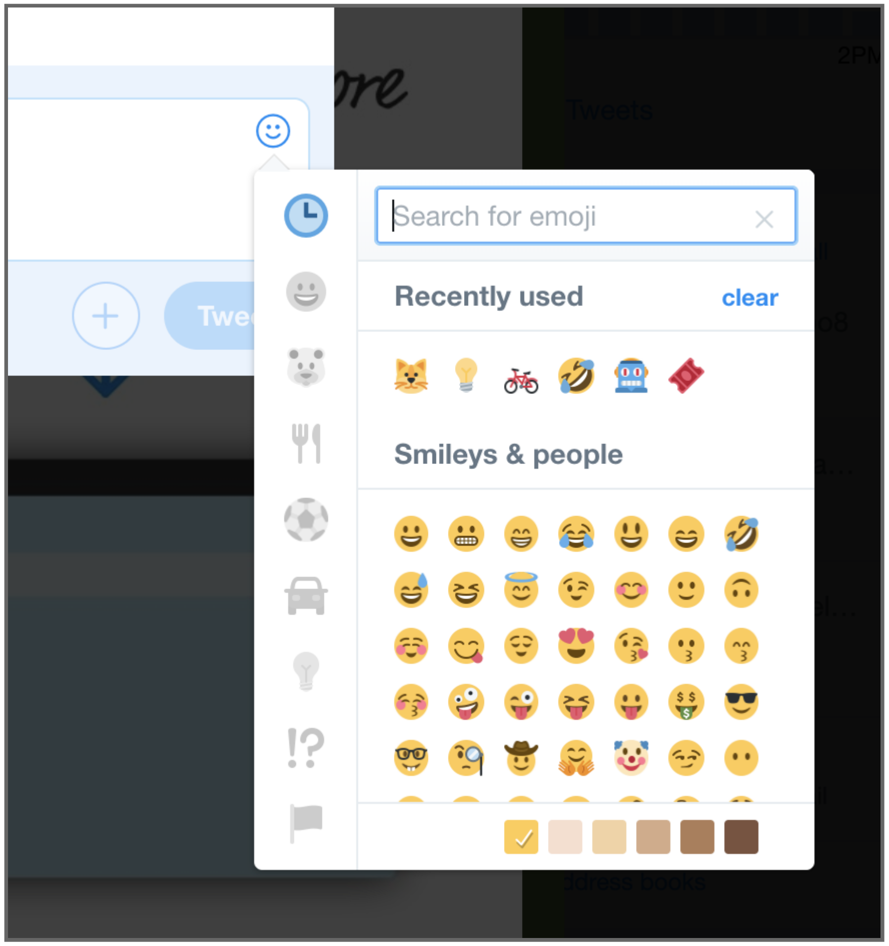
Using Emojis in HTML, CSS, and JavaScript

by **kirupa** | filed under **UI Stuff**

From its humble beginnings in 1999, Emojis are all the rage these days. It's no longer something that only people half our age use to communicate. You and I use them all the time, and almost every chat or messaging-related app under the sun provides great support for it:

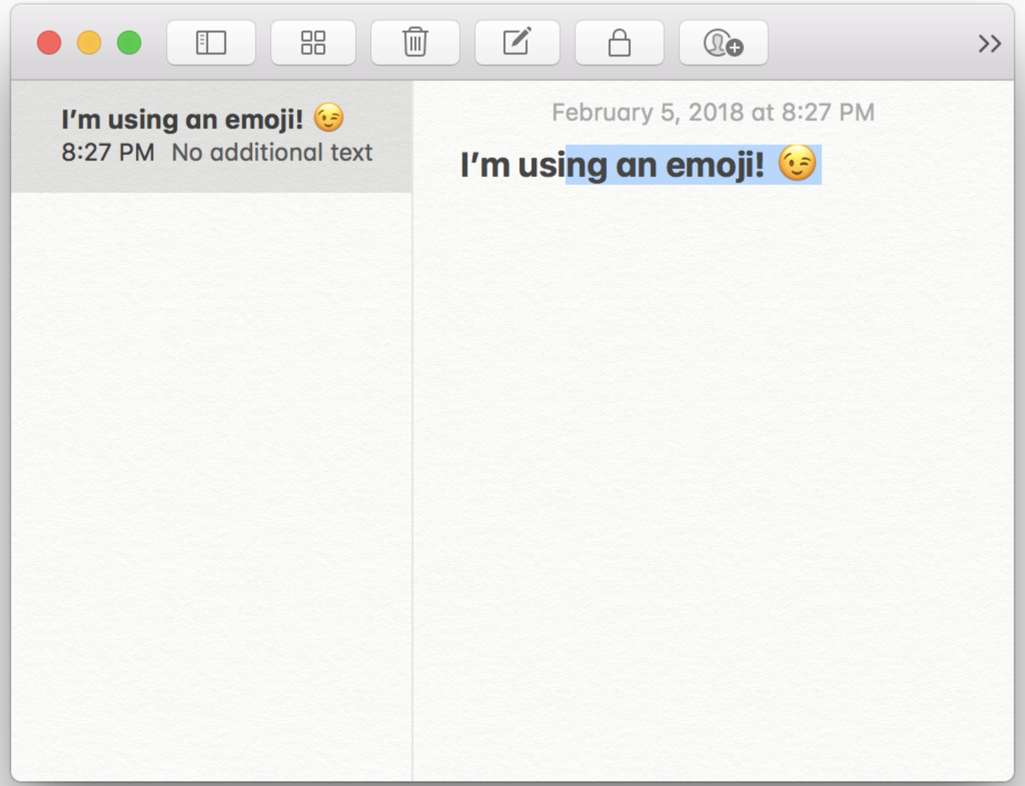


For everyday users, emojis are great. They are fun and easy to use. For us web developers wanting to use emojis in our HTML, CSS, and JavaScript, the story is a bit different. There are a few hoops we need to learn how to jump through, but don't worry. This tutorial will help you master all of this hoop jumping like a pro!

Onwards!

What are Emojis Exactly?

We already know that emojis are these tiny colorful icons. While this may give you the impression that they are images in the traditional sense, they aren't. They are more like the letters, numbers, punctuation marks, and weird symbols that we tend to bucket as text:



Towards the end of this tutorial, I go into much greater detail on what emojis are and some of the details under the covers that makes them work. For now, just know the following:

i. They are just characters

ii. You can select them, copy them, paste them, adjust their size, and so on

iii. They have a more primitive numerical representation that you can use to get them to display

To say we just scratched the surface in understanding emojis is an overstatement, but this is enough for us to get started. It's time to see emojis in action inside our web documents!

Emojis in HTML

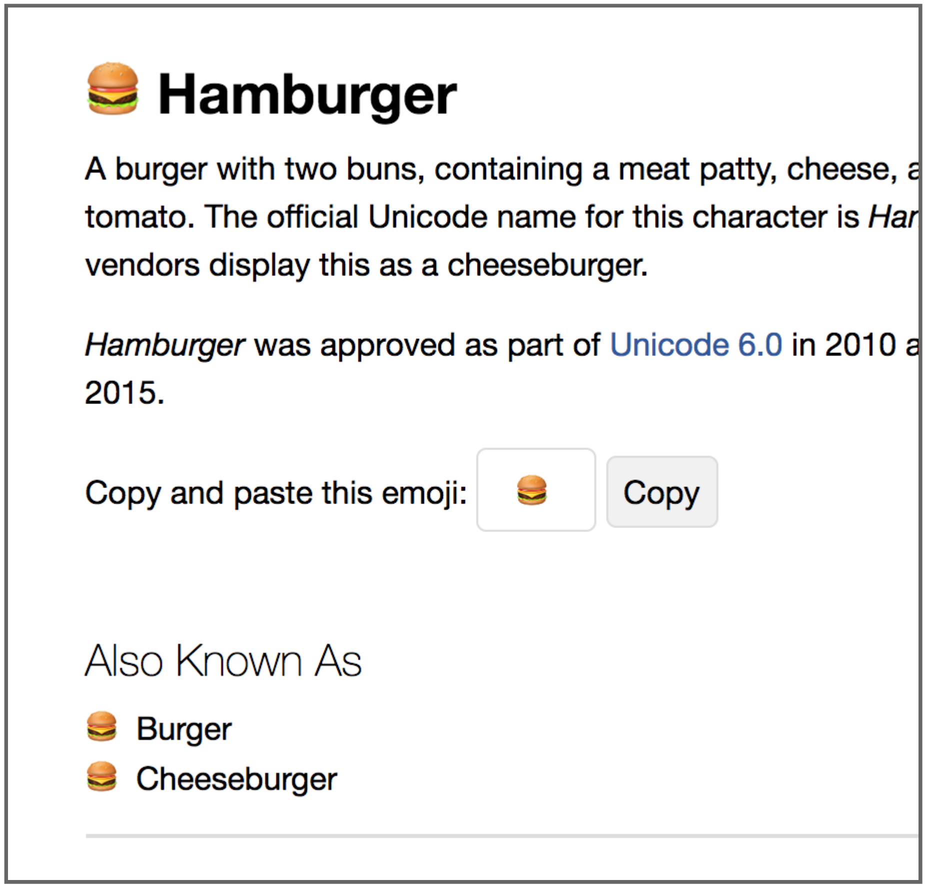
To use emojis in HTML, the first thing we need to do is set the document's character encoding to UTF-8. This ensures our emojis display consistently across the variety of browsers and devices your users may be running. Doing this is simple. Inside your `head` tag, be sure to specify the following `meta` tag:

```
<meta charset="UTF-8">
```

Once you've done this, now comes the fun part of actually getting an emoji to display. You have two ways of being able to do this, each with a varying degree of funness. One way is by using the emoji directly in your HTML. The other way is by specifying the emoji via its *primitive numerical representation*. We'll look at both of these cases.

Using the Emoji Directly

The easiest way to display an emoji involves simply copying and pasting. You just need an app or web site that allows you to copy emojis in their native, character form. One great place for doing that is Emojipedia. You can use Emojipedia to search or browse for whatever emoji you are looking for. Once you've found your emoji, there is a section where you can easily see and copy the emoji:



🍔 Hamburger

A burger with two buns, containing a meat patty, cheese, a
tomato. The official Unicode name for this character is *Har*
vendors display this as a cheeseburger.

Hamburger was approved as part of Unicode 6.0 in 2010 a
2015.

Copy and paste this emoji: 🍔 Copy

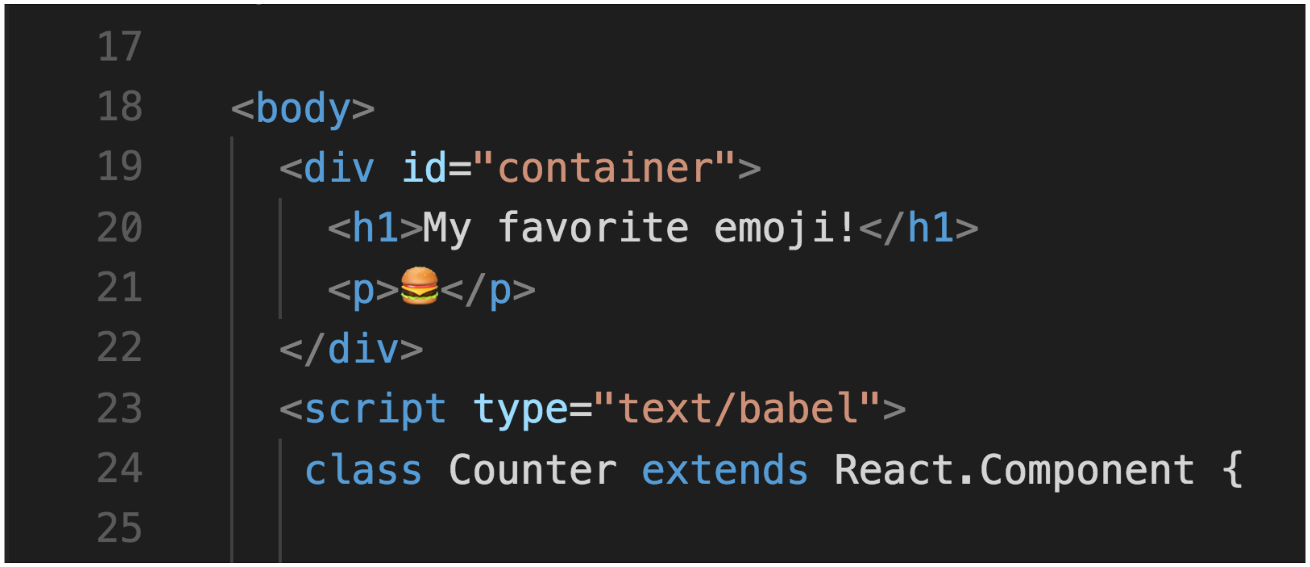
Also Known As

🍔 Burger
🍔 Cheeseburger

Once you have copied it, just paste it in its intended destination in your markup:

```
<p>🍔</p>
```

Because emojis are treated as text-based content, you can paste them almost anywhere in your document where text is supported. Now, if this is your first time seeing emojis randomly appearing inside your text-based code or your code editor, it will look a bit strange:



```

<body>
  <div id="container">
    <h1>My favorite emoji!</h1>
    <p>🍔</p>
  </div>
  <script type="text/babel">
   class Counter extends React.Component {

```

Your traditional text-only environment where you've written your markup all these years will suddenly have something visual in it. Don't worry. It's cool. When you preview your HTML document in your browser, everything will still work.

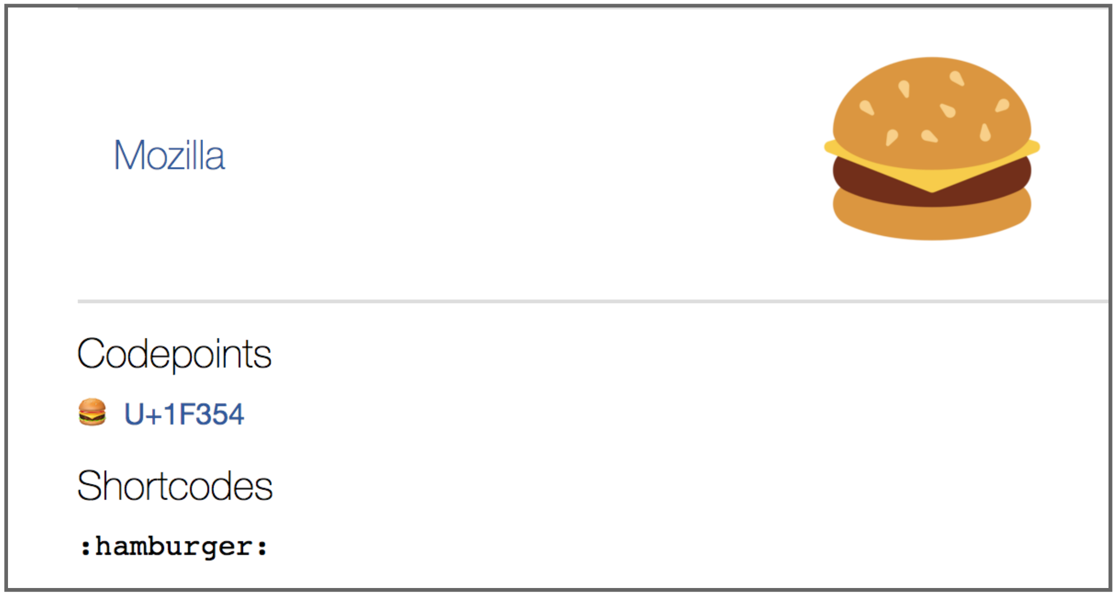
Before we wrap our look at emojis in HTML, let's talk about accessibility. Emojis are ultimately visual artifacts, but they are represented as text under the covers using elements like `p` and `span` that are semantically ambiguous in this case. Screen readers and related tools may not interpret what the emoji is trying to signify properly, but the solution is simple. To use emojis in an accessibility (***a11y*** as the cool kids call it!) friendly way, set the `role` and `aria-label` attributes on the element that you are using to reprsent your emoji:

```
<p role="image" aria-label="hamburger">🍔</p>
```

With this change, you still get to use emojis in HTML and be accessibile at the same time. That's a win, win,...win! Thanks to Eva Larumbe for pointing this out.

Specifying the Emoji Codepoint

If specifying the emoji directly doesn't work, there is a less fun path you can take. You can use the emoji's numerical representation and specify it in your markup instead. If you scroll down in Emojipedia for any emoji, you'll see the numerical representation (more formally known as a **codepoint**) displayed:



Codepoints

🍔 U+1F354

Shortcodes

:hamburger:

For the hamburger emoji, the codepoint is **U+1F354**. To specify this emoji in HTML using the codepoint, we have to modify the value a bit. Add the **&#x** characters, remove the **U+1** from the beginning of the codepoint, and just add the remaining digits from the codepoint as part of any text element.

Here is our hamburger emoji in codepoint form:

```
<p>&#x1F354</p>
```

When you preview your document in your browser, you'll still see the hamburger emoji displayed correctly...despite it looking strange in our markup compared to the copy/paste solution we looked at earlier.

Emojis in CSS

> You can totally use emojis in CSS. The same tricks we saw for emojis in HTML will work with only some slight modifications. You can specify the emoji directly:
>
> ```
> h1::before  {
>   content: "🍔";
> }
> ```
>
> You can also specify the emoji by setting the codepoint:
>
> ```
> h1::before  {
>   content: "\01F354";
> }
> ```
>
> How you specify the codepoint is a bit different than what we saw for HTML. All you have to do is remove the **U+** from the unicode endpoint and add the `\0` (slash zero) characters just before it.

Emojis in JavaScript

The last thing we will look at is how to use emojis in JavaScript. The approach of using it directly will work here as well. Just make sure to treat the emoji as text:

```
document.body.innerText = "😀";
```

To use an emoji via its codepoint value instead, we have to pass them through the `String.fromCodePoint` method. This method takes a codepoint value as its argument:

```
document.body.innerText = String.fromCodePoint(0x1F354);
```

What gets returned is the character at that codepoint location. How you specify the codepoint is different than both HTML and CSS. If the codepoint is **U+1F354**, replace the **U+** with **0x** (zero and x) before passing it in. That's it. If you want to go further, since you are in JavaScript, you can do all sorts of JavaScripty things. You can have an array of emojis, you can dynamically generate them, and so on:

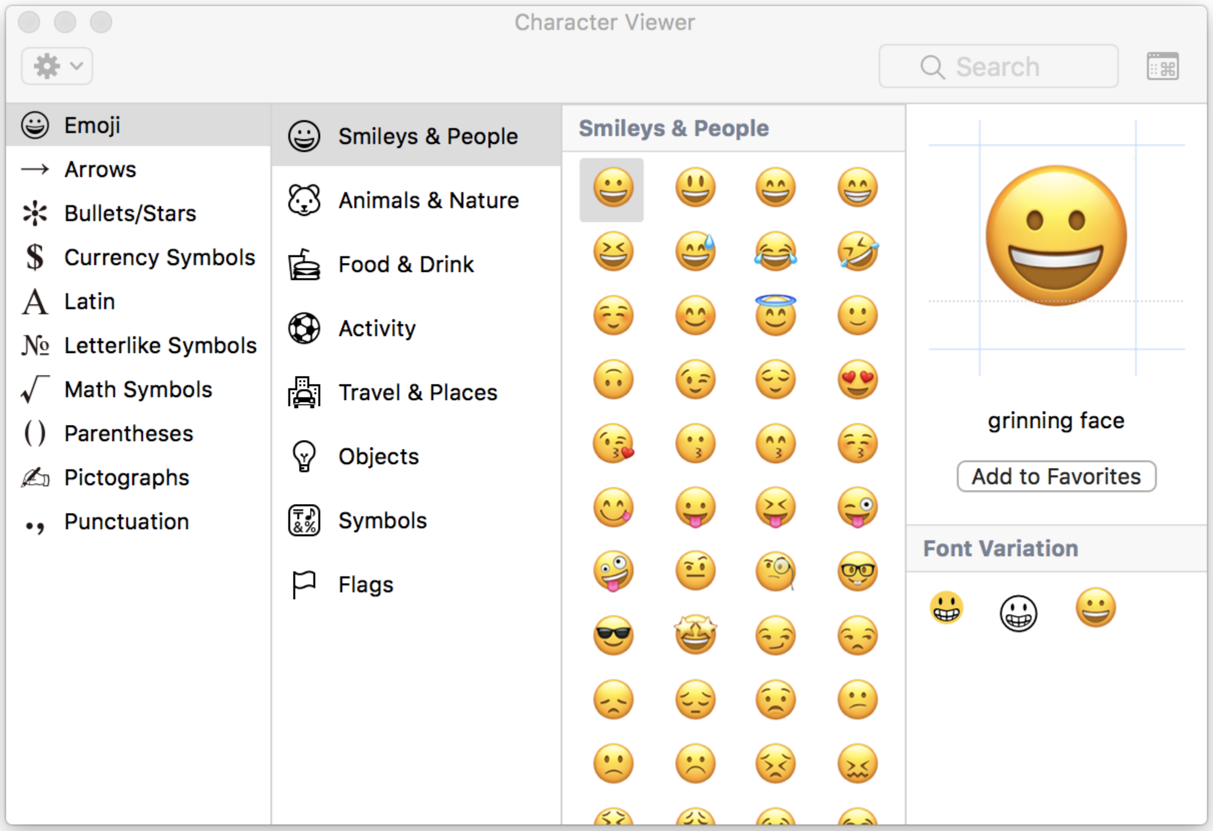
```
var emojis = [0x1F600, 0x1F604, 0x1F34A, 0x1F344, 0x1F37F, 0x1F363, 0x1F370, 0x1F355,
              0x1F354, 0x1F35F, 0x1F6C0, 0x1F48E, 0x1F5FA, 0x23F0, 0x1F579, 0x1F4DA,
              0x1F431, 0x1F42A, 0x1F439, 0x1F424];
```

If you are curious to see a working example that uses emojis defined in JavaScript, check out my Koncentration game. The Github repo (especially Board.js) contains everything you need to see how the game is tied together.

Some Emoji Details

We've glossed over what emojis really are and what purpose codepoints serve. To start from what we said earlier, emojis are just characters like all of the text that we type. That is often a confusing thing to understand. A part of the reason for this confusion is because we tend to think of characters as just what our keyboards support. Here is the problem: ***What our keyboards allow us to represent is a very tiny percentage of the overall set of characters that is available to us.***

If you want to see this for yourself, you can see all the characters your operating system supports by using Character Map on Windows or the Character Viewer on Mac:



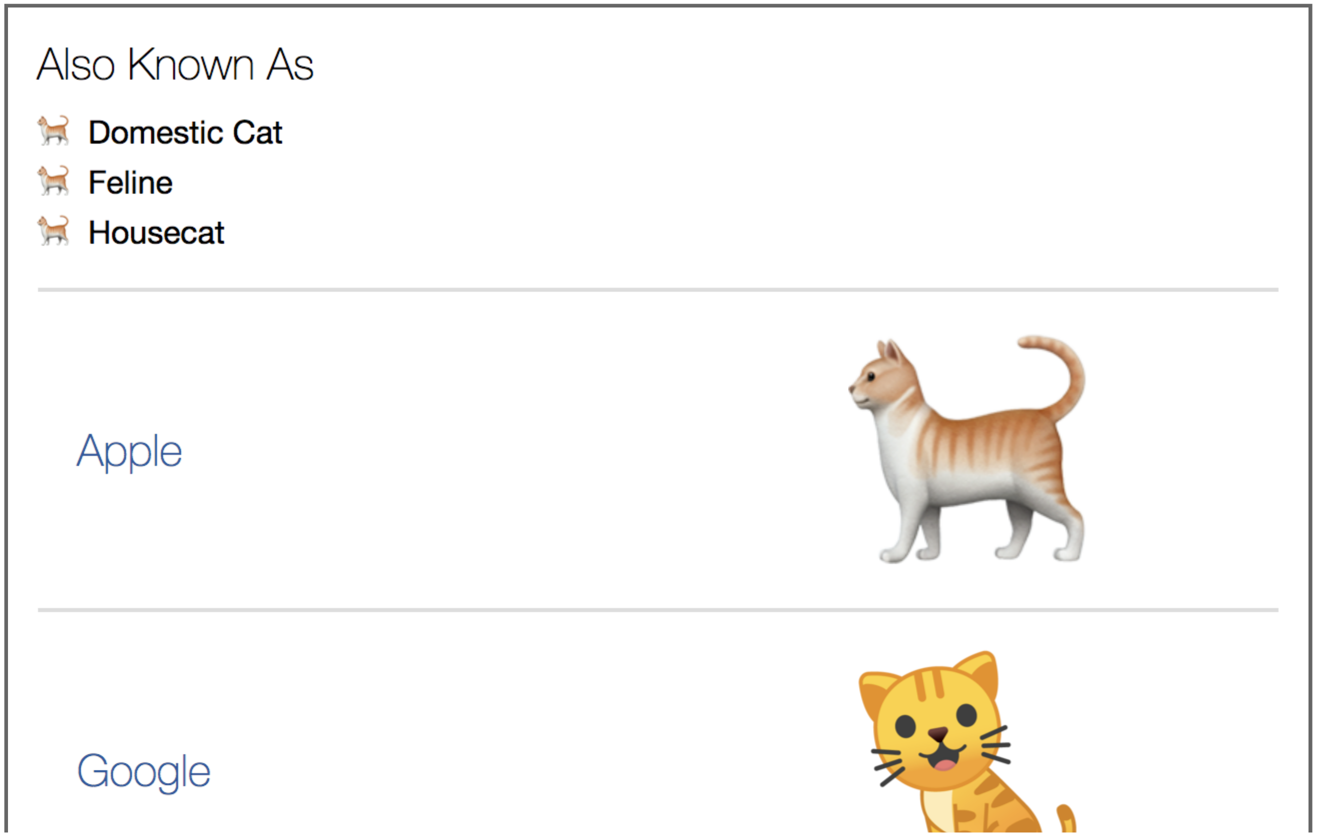
Notice that you have many categories of characters that go beyond just the usual things we see on a typical keyboard. If we had to have a keyboard to support every character our operating system supports, it would need to be at least...three times bigger than the usual keyboard.

Ok! Let's go one level deeper. We saw that the emojis we wanted to use had a bizarre numerical representation. As it turns out, ALL characters we use have the same bizarre representation under the covers as well. On our screens, we may see letter characters like **A**, **B**, **C**, **D**, **E**, **F**, **G**. Under the covers, these characters look like the following: **U+0041**, **U+0042**, **U+0043**, **U+0044**, **U+0045**, **U+0046**, and **U+0047**. We already said that this representation is known as a codepoint, but the more precise term is **Unicode codepoint**. This detail is important to know about, for Unicode is an

> industry standard for ensuring the text you see on your screen is the same on another screen somewhere else - regardless of language, locale, system capabilities, operating system, and so on. A codepoint is the most basic unit of representing information in Unicode. A series of codepoints represents characters and text. Phew!

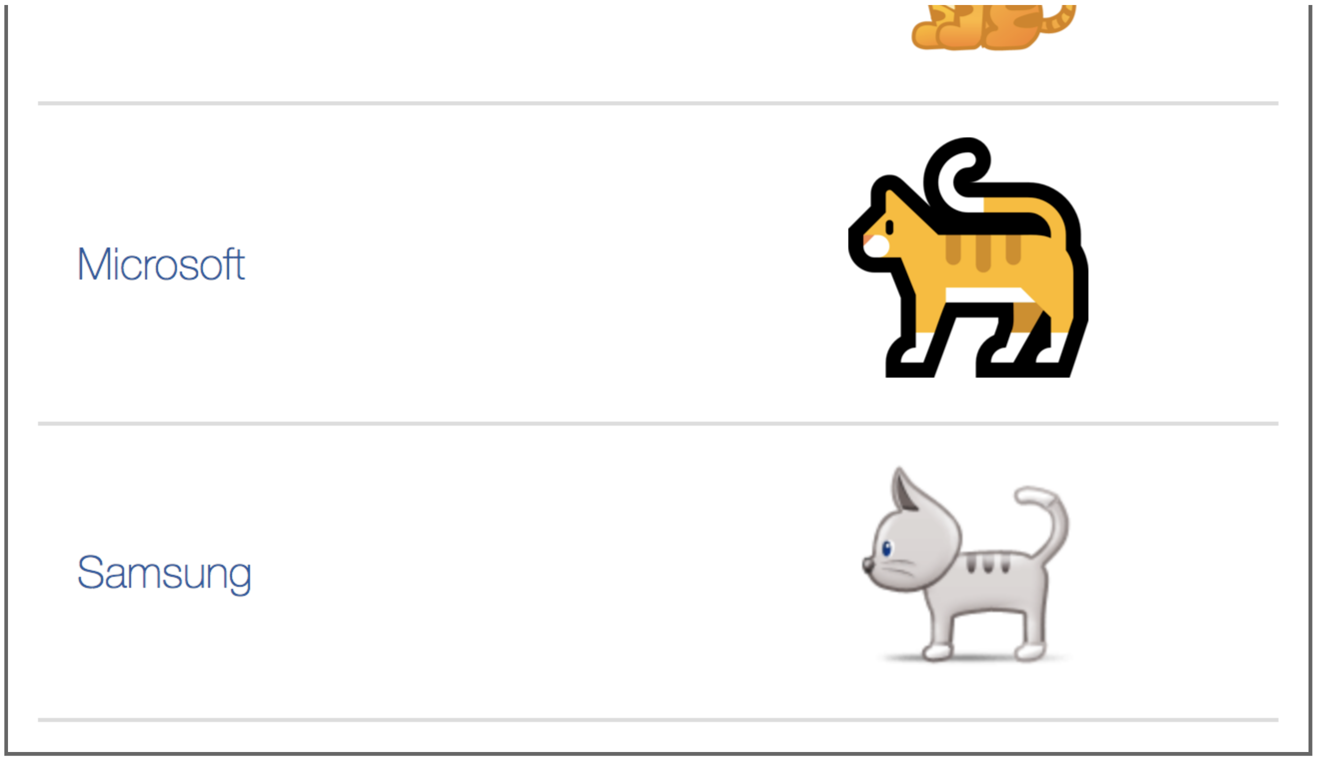
Conclusion

If you can get away with it, copying and pasting emojis is the easiest thing you can do across HTML, CSS, and JavaScript. There will be situations where you can't do that, so you the fallback approach involving codepoints. There is one last thing that may be important to you. Because emojis are native to the app or platform you are on, emojis can look different for different users:



Also Known As

- 🐈 Domestic Cat
- 🐈 Feline
- 🐈 Housecat



Microsoft	
Samsung	

The Unicode standard ensures that the appropriate codepoints represent, in this case, a cat. Each implementer has full creative freedom in interpreting that as they wish. For some developers, this inconsistency isn't desirable. What they have done instead is re-create the emojis in SVG or PNG form so that they can ensure consistency. An example of someone who does that is Twitter! We used their emoji picker screenshot to start this tutorial off at the beginning, and every emoji you see there isn't from our operating system or platform. It is from the really awesome Twemoji project. There are many emoji libraries out there that you can use, so use whichever one you like if the native emoji support isn't what you are looking for.

If you have a question about this or any other topic, the easiest thing is to comment below or drop by our forums where a bunch of the friendliest people you'll ever run into will be happy to help you out!

Kirupa

THE KIRUPA NEWSLETTER

Get cool tips, tricks, ~~selfies~~, and more...personally hand-delivered to your inbox!

email address

SIGN ME UP

(View past issues for an idea of what you've been missing out on all this time!)

GOT A QUESTION?

6 replies

prg9 Registered User Feb '18

¯\_(ツ)_/¯

1 reply

prg9 Registered User Feb '18

（ ^_^）o自自o（^_^ ）

prg9 Registered User Feb '18

Fun Article K-man.

e_v Registered User Feb '18

Thank you. Very cool. 🧙

Thomas_Stokes 31 Jan

😂

Horacio_Ramirez ▶ prg9 24 Apr

¯\_(ツ)_/¯

Continue Discussion

BACK TO TOP