What's a web app?

How the apps you pretend to use at work...work

Justin
Feb 26, 2020

If you like this, subscribe and get some more:

Subscribe now

The TL;DR

A web application is a series of interactive web pages put together and accessed through your browser.

- A **web page is a group of files** that you download from a server
- Some websites look the same for everyone, but **dynamic sites like Twitter change their content** under different circumstances
- Web apps like Google Drive are a series of dynamic web pages, usually behind a login

Any software you use on the internet is probably a web app, and almost every application you use follows the same general pattern. So read this!

(I didn't understand what a web app really was until years after I learned how to code.)

First: what's a website?

The key to understanding web applications is understanding websites, because apps are just big, self contained sites. And contrary to popular belief, a website is not actual magic.

> 🔮 **Dependencies** 🔮
>
> To get the most out of this post, you'll want to understand the basics of cloud (what web servers are), as well as the frontend / backend distinction and what's powering our apps.
>
> 🔮 **Dependencies** 🔮

When you're loading a site, you're actually *downloading files* from a web server in the cloud. That's it! The URL in your browser tells the web server which files to send you, and then your browser displays them to you. There are **3 basic types of these files** that create what you see on your screen:

1. HTML: the **text and structure** of the elements on the page (a header, a paragraph, a table)
2. CSS: the **styling** of the words and structures that you defined in your HTML
3. JS (Javascript): the **interactivity** of your HTML-defined elements

If you're reading this in your Gmail app, you're interacting with these 3 types of files **right now**. There's an **HTML file** defining the structure of what you see: the text of the email, the tabs on the left, and the search bar on the top. The **CSS** adds fonts, borders, coloring, and sizes. And the **Javascript** changes tab color when you hover, adds little tooltips, and animates transitions. In practice websites can have hundreds of these HTML and Javascript files, and a few CSS files too.

Contrary to the authoritative tone of this piece of writing, you don't need to just take my word for any of this. Load a website on your computer (this won't work on your phone) and save it (File → Save Page As). Head over to your downloads folder (or wherever you saved it) and open the file, and you should see something pretty close to what you'd get if you'd put in the URL. Even if you disconnect from the internet!

When you navigate around a site and click on to a different page, you're loading a *different* file (that's why the URL changes!). A modern website is just a collection of these different files, connected to each other through links (embedded in buttons, arrows, you name it), being sent to your computer whenever you request

buttons, arrows, you name it), being sent to your computer whenever you request them. Where things get complicated is *how* these files get built.

Dynamic web content

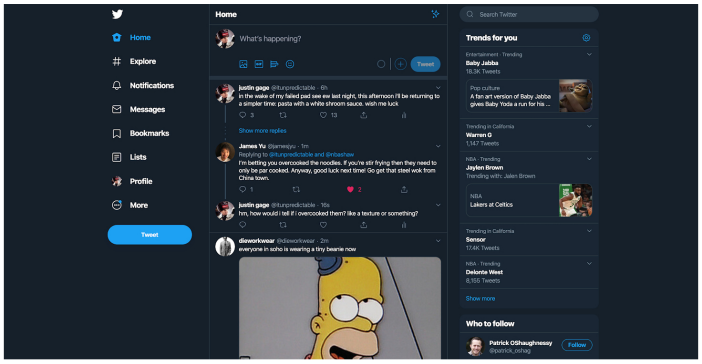
A lot of websites are pretty much the same for everyone who loads them: the front page of the New York Times looks pretty standard, for example. But when you load Twitter, your feed is completely different from what someone else might see on their account. Plus, new tweets seem to show up all the time, without you reloading the page. How is all of that happening?

Websites can **load and display content *dynamically***. Twitter doesn't send the same files to everyone: they customize what they send you based on who you're logged in as, who you follow, what words you've muted, and other stuff like that. Here's what's happening behind the scenes:

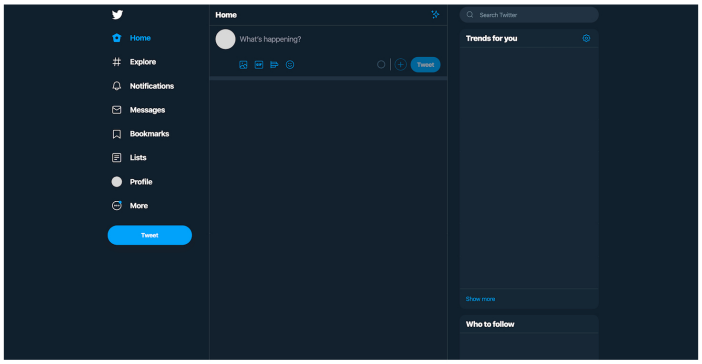
- You tell the Twitter web server to send you the files for your homepage by putting *twitter.com* in your browser
- Twitter checks who you are – you're logged in as yourself (nice), with a username
- Twitter goes to their database and generates a feed of tweets based on your information
- Twitter puts that content in the HTML file, and sends it over to your computer

(To get into the nitty gritty, a bunch of times they'll send you the files and then add the dynamic content, but that's for another time)

On most web pages, some content is *static* and some is *dynamic*. Think about your Twitter homepage:



A lot of things here don't change much. The order and the text of the tabs on the left, the logo, the search bar – all of this is the same for everyone. But what shows up in the feed, the little picture of me next to "profile" on the left, and the trending tweets on the right – those are all dynamic. If I had to guess, this is what the Twitter page looks like *before* the dynamic content gets put in:



Once you log in, Twitter adds your profile photo (they stored it in their database) to the two little gray circles, and populates your feed with the right tweets and trending topics based on what they think you'll want.

The Google Drive web app

That's all a web app really is: it's a series of dynamic web pages. Every piece of

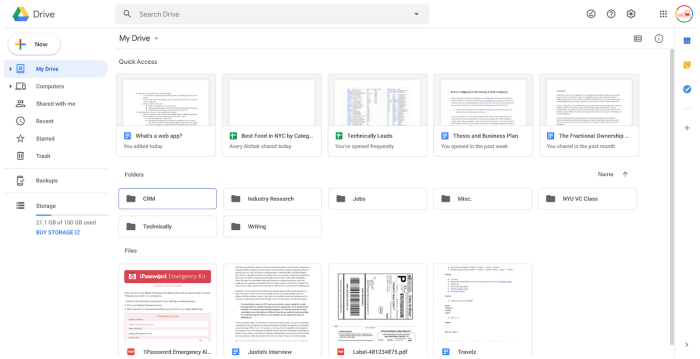
software you use follows this paradigm. Let's take a look at a great example (if I might say so myself) – Google Drive – and walk through what happens when you're forced to use it under the guise of productivity.

1. *Login*

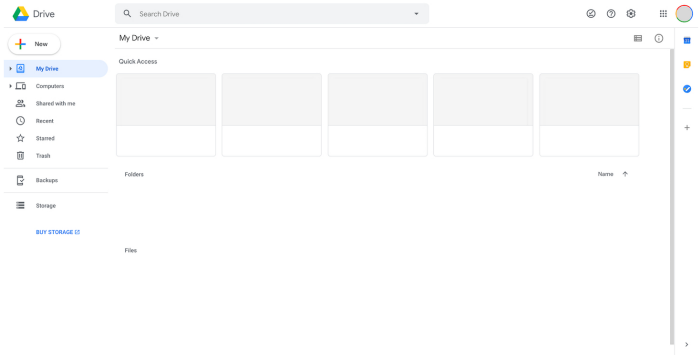
If you haven't logged in in a while or your preference is to require credentials on every page load, Google Drive will prompt you for your login information. When you set up your account, you chose a username and password; Google stored that information in a database. As you click "login" Google takes the username and password you supplied, and checks it against what they have in their database: if it matches, they let you in.

2. *Home page*

Everyone's Google Drive homepage looks different, because like Twitter, almost the entire page is dynamic. Now that you've logged in and Google knows who you are, they head into their database and find all of the content (Google Docs, Sheets, and Slides) that you've created in the past. They then format that content and slide it into the HTML that shows up in your browser.



Without the dynamic content served from the backend, the page would probably look something like this:



3. *Creating and editing a spreadsheet*

A spreadsheet in Google Sheets is just a bunch of data, and Google is *storing and retrieving* all of that data constantly. When you create a new Sheet, Google records that in their database. When you edit a cell, create a formula, or create a new sheet, Google records that too. All of the information sits in Google's servers, and is sent and displayed to your computer; which is why you can access this from any other computer too (cloud!).

> ⛓ **Related Concepts** ⛓
>
> There's a lot to explore around the details of how web apps get built and served, like client side rendering (React, Angular, Vue), HTTP, caching, CDNs, and lots of other stuff. We'll cover those in their own posts. One day.
>
> ⛓ **Related Concepts** ⛓

"Web app" in conversation

> *"I like the Desktop version of Excel better. The cloud web app is kind of janky."*

Excel has a web app that works in the cloud, but the version that runs on my laptop is easier to use and more reliable

laptop is easier to use and more reliable.

> *"The outage brought down our web app, so we got paged."*

Our cloud provider had a service outage, so our servers weren't working: they weren't able to send users the files they needed, so our app went down.

> *"We started using React a few months ago for our web app"*

We've been using the React framework (ooh! aah!) to build our web apps and render dynamic content to our users.

Terms and concepts covered

```
Web application
```

```
HTML / CSS / JS
```

```
Dynamic content
```

Further reading

- People write web app backends in all different languages, but Javascript is really having a moment
- Rendering dynamic content to your users can happen on the server side or on the client side, and the latter has been getting a lot of love through React and Vue
- It's much easier to get a website up and running than it used to be, and static site builders like Jekyll and Hugo are hitting the mainstream

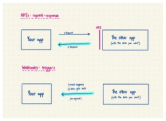
If you liked this, jump in and subscribe!

Justin Jan 4 8

What's DevOps?

IT has a cool new name

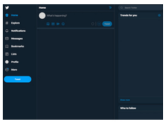
Justin Jan 5, 2021 34



What are webhooks?

Triggered

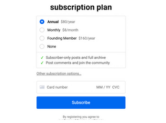
Justin Sep 13, 2021 28 5



What's Headless E-Commerce?

We may be running out of names

Justin Nov 2, 2021 20 7



I built a (basic) Substack clone in a month

This was probably a waste of my time

Justin Sep 2, 2020 50 4

See all ›