

Making Websites and Putting Them Online

There are *loads* of different ways to put your work online, and trying to figure out what will work for you can be quite overwhelming. The good news is, depending on what you are trying to do, most approaches will work! The approach you take will depend on what kind of website you are trying to make, the tools you are comfortable with, and the software you are using to make your site.

To put a website online, you need to answer 2 distinct (but linked) questions.

1. **what kind of website am I making?** This will determine the tools and frameworks you will use. If you have a project you've already made then you have pretty much answered this question.
2. **what are my options for hosting?** This will include budget constraints, whether the site needs to be editable, and considerations like longevity/stability.

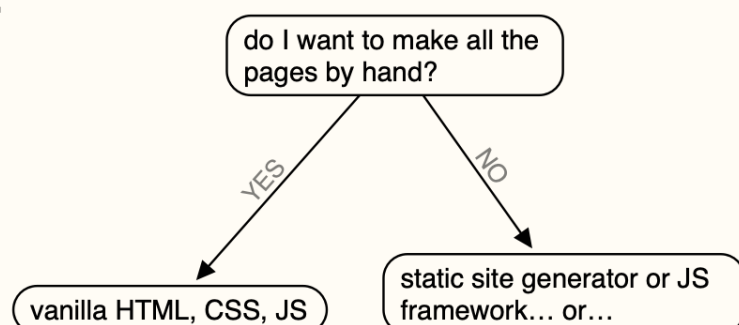
1. What type of website am I making?

One of the hardest things about beginning web development is having an idea of what is and isn't possible with different web technologies. This can take some time to get a sense for, but the core divide with most websites is -- do you need to permanently store changing information? (this is different from just being interactive). This guide assumes you're going to write your own code (out of wholesomeness), there are also ways of doing these things without that step.

If you're not storing information, you most probably will be making a static site. Most websites you might make in the CCI fall under this category:

WHAT KIND OF WEBSITE AM I MAKING?

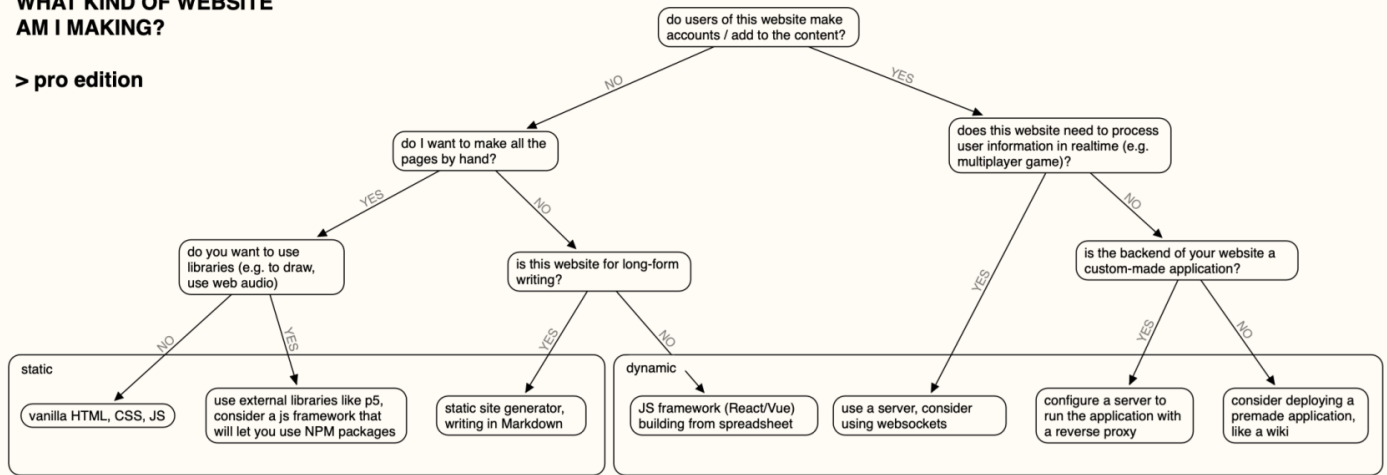
> mini edition



...or! If you're interested in professional web development, or you want to do something funky (maybe you want to get live information from a sensor, or host a multiplayer game or a chat), you might want to look into some more advanced approaches.

WHAT KIND OF WEBSITE AM I MAKING?

> pro edition



Note that none of these end points are what you *have* to do -- there's still lots of different ways to do things, and also it's most likely that you will want some combination of what's here if you are doing anything more complicated.

Static Websites

'Static' websites comprise most of the websites you might make at the CCI. They're called static because broadly, they don't change permanently -- they might be interactive, but every time you load them they should look the same. These can be made using 'vanilla' HTML+CSS+JS, using libraries (like p5, jQuery), using Javascript frameworks.

we love vanilla JS

You will hear the phrase 'vanilla' used to describe websites that are made using just HTML, CSS and Javascript. These are the building blocks of the web, it's important to know how to use them! You can make really good websites just using these -- their downside is when you need to make lots of pages (it can get laborious writing HTML), or when your site needs to be edited by people that don't code.

an aside: pure html There's been some nice projects in recent years to revive the practice of making sites *just* using vanilla HTML and CSS. A nice one is html.energy.

JS Libraries

There are lots of ways to extend the functionality of your code without needing to change the structure of your site. Libraries contain functions written by other people that make it quick and easy to do things (like drawing, interactivity) that would otherwise be laborious to write. Here are some common ones that you may already have met:

- [jQuery](#) -- has been around for decades, very useful for interactivity

- [p5.js](#) -- the JS implementation of processing, used a lot in the CCI. Very useful for simple drawing, audio and interactive applications
- [tone.js](#) -- really useful site for JS audio

JS libraries can be included in static sites either through using CDN links or local files. Make sure to include the library in your HTML before other Javascript that uses it! If you're getting to the stage where you are using lots of libraries and finding them hard to manage, you might want to consider using a JS framework that can use node modules (more on this below).

Static Site Generators

These are really great if you want to make a blog, or a site that has a lot of text. My [personal website](#) is made using the static site generator Jekyll, and I also used to make my [teaching websites](#) that way. Jekyll is great but a bit old-fashioned -- if you're making one today most people use Gatsby.

Static site generators *generate* HTML, CSS etc from an easier-to-maintain format, often Markdown or JSON. It makes these sites much easier to add to and maintain, and allows you to use some templates to lay out pages.

Somewhat confusingly, static site generators themselves can often be written in a different programming language -- Jekyll is written in Ruby, so to use Jekyll you also need to install Ruby. Gatsby is nice as it's also written in Javascript!

JS Frameworks

These days, lots of websites are written using Javascript frameworks. The most common of these is [React](#), but there are lots of options (Vue is also popular). These frameworks often work along the idea of webpages being made of reusable, interactive components. This can be really helpful in building sites with complex interfaces and changing information.

These also make use of the Node Package Manager, which is used to manage the libraries used by a site. In general, frameworks can make developing complex things a lot easier!

External APIs

Lots of websites draw information from other sources on the web. There's no limit to the number and types of API your site can use. In the case of CMSs, all of the information from your site might be drawn from an external API.

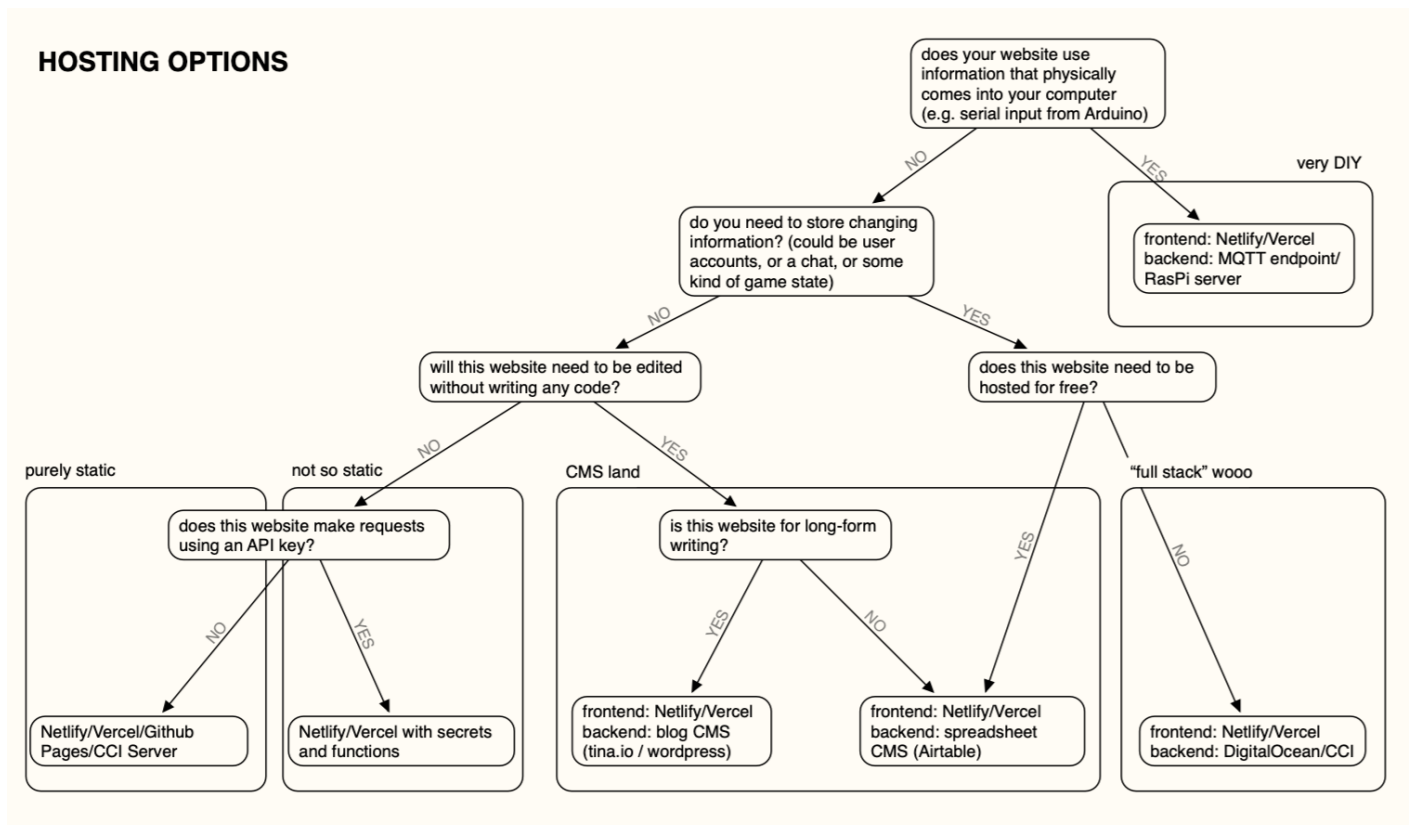
"The backend"

If you're wanting to persist information and update with a high degree of reliability, chances are you are going to want to use a database. There are a bunch of services that allow you to do this

without having to run your own server (e.g. AWS), but it can also be very useful to know how to do this yourself.

Hosting

The biggest part of putting anything online is this -- how do you 'host' your site, so it stays up and people on the internet can find and use it. The hosting method you use will depend on the kind of site you are making, who needs to edit your site, whether you're needing to store information in a database, and whether you're doing anything that depends on physical infrastructure that you've built.



The most common ways you will host websites in the CCI (if you take a web class) are either on the CCI server, on a web editor like p5, or on Github Pages.

Hosting Static Sites

These are the easiest kinds of sites to host, and there are lots of different options! At this point in web history, you should basically never have to pay to host a static site (unless you have like 10million visitors), though you will still have to pay for domains.

All static sites are comprised of files (HTML, CSS and JS files) that are sent to visitors to the site (the 'client') when they

Web editors (e.g. p5 JS, Glitch)

Online coding editors like Glitch and p5 can allow you to quickly share work with others. You might use these in class, and these can be a good way to set up a site other people can see without having to do any other work! However, often these have quite basic support for using e.g. custom domains and can take some time to load, and don't make a good permanent solution.

The CCI Server

At the CCI we maintain a web server that students and staff can use to host their work. By default, unless you take a class that requires it, you will not have an account on this -- but you can message the [#technical](#) channel on slack and we will make you one.

We have a separate wiki page with instructions for [connecting to the CCI server](#). You will use a protocol called FTP -- the File Transfer Protocol. This sends files from your computer to the server directly, and used to be the main way people put things online.

Github Pages

Using Github Pages is popular in the department, and depending on the class you take, your teacher might get you to use [Github](#) or the [git.arts server](#). These look very similar, the main difference is that git.arts is managed by us, and you will need to use your CCI account to connect to it. For the purpose of this tutorial, we will use Github. We have tutorials on the wiki for [using these](#).

Github Pages gives you a way to host a site that is saved as a repository on Github. You can create this either as a git repository on your computer, or directly edit the files in Github itself. The former is a good idea as it also allows you to test the site out, and make the most of the versioning tools.

Vercel and Netlify

Vercel and Netlify are free (for regular use) web hosting services that integrate with Github. They have slightly more features and options than Github Pages, though for sites made with HTML, CSS and JS you will not notice much of a difference. Like Pages, these are best used as git repositories, but also allow you to drag and drop files.

I use all of these three (I tend to use Vercel these days) to host static sites. Vercel and Netlify are normally better than Github Pages for hosting sites made with frameworks like React and Gatsby.

Hosting Dynamic Sites

Dynamic websites are websites where information on the site is stored and changed, such that it changes for all users of the site. Examples include multiplayer games, websites with user accounts, wikis, and websites that display the state of some external process (like a weather station or a sensor). These can require quite a lot more steps to make than static sites, but in doing so you learn a lot more about the web!

'Serverless' Processes

Between 'totally static' and 'fully dynamic' websites are sites that are mostly frontend, but will pull information from other sources -- for example, through an external API, or potentially a lightweight CMS like Airtable or Tina. The work of populating this site with information can basically all be done in the frontend, but one thing you need to be careful with is publishing the 'API key' used.

Netlify has a service called functions that can be really useful for this -- these run on the backend of your application, so the client can't see the API keys.

If you don't want to go to all this effort, lots of services (including Airtable) will allow you to make 'controlled access' keys, that only have read permissions. This can be fine if you are only ever reading from a source rather than writing to it directly.

CMS Options

If you make a site for someone in a professional context, chances are that if they have any desire to edit the site, they won't want to edit the code directly. In this instance, you will want to make a CMS, or 'Content Management System'. Normally, what this will look like is some kind of list of pages or files (if you're using a text-based editor like Wordpress or Tina), or a spreadsheet like Airtable.

I like to use Airtable as a CMS for websites, as it's free and has a really good API. I also tend to make websites for artists, who have mostly images and small amounts of text: the only time I wouldn't do this is if someone wanted to host lots of writing.

The screenshot shows the Airtable 'Projects' interface. The left sidebar contains a 'Create...' menu with options: Grid (selected), Calendar, Gallery, Kanban, Timeline (marked 'Team'), List, Gantt (marked 'Team'), New section (marked 'Team'), and Form. The main area displays a grid view of 47 projects. The table has columns: Name, Date From, Date To, Venue, and Funder/Producer. The data is as follows:

	Name	Date From	Date To	Venue	Funder/Producer
1	Taken by the Water	2023			A Two Cats Pictures
2	Adaptation, Enough Alrea...	2023	26/8/2023	Vegas Nerve	Vegas Nerve
3	Next of Kin)/2022			
4	Green Satanic Thrills	2023		Camden People's Theatre	Camden People's T
5	her bed // dream wake pie...	2023		Sadler's Wells (The Lilian ...	Second Movement
6	All The Little Lights	2019	17/8/2019	Tristan Bates Theatre	
7	Camellia	:021	17/7/2021		The Japan Foundati
8	Sushi Delivery)/2019	18/10/2019	Low Profile Studios	
9	the accident did not take ...	2019	1/3/2020	Pleasance Theatre Edinb...	HOME, ARC Stockto
10	Incubator	:019	10/5/2019	HOME	HOME
11	Four Seasons of Buenos ...	:019	5/4/2019	HOME	HOME
12	Experiments with Art and ...	:2018	18/5/2018	Birkbeck University of Lo...	Birkbeck School of /
13	The Government Inspector	2023	20/3/2024	The Lyric Hammersmith a...	The Lyric Hammersr
14	La cambiale di matrimoni...	2023	19/5/2023	Royal Academy of Music	
15	The Rake's Progress (Ob...)/2022	27/11/2022	Royal Academy of Music	
16	Sparkplug (Assistant)	2019	13/4/2019	UK Tour	Box of Tricks
17	...ntance of Being ...	:019	15/6/2019	Albert Halls Bolton	Bolton Octagon The

With a CMS, you fetch the information from the API, and then use it to populate the site you have made. You *can* do this with vanilla JS, but frameworks like React come with a lot more tools that can make this process easier.

Using a Server

If you want to run a custom-made application, you might be at the point of needing to use a server. A use-case for this might be, for example, a Python-based multiplayer game that has a web interface, or any continually-running process where all clients need to see the same state. Learning to use a server can be a big learning curve in terms of the skills you might need, but is also an amazing way to learn more about how computers work.

Server programming is a huge topic, but I'll list some common tasks you might want to do! If you're getting into server programming, Digital Ocean's tutorials are a great place to start, and I've linked them here.

Domains, DNS and SSL

The last part of hosting a website is often adding your own domain to it. Providers like Github Pages, Netlify and Vercel will allow you to modify the domain they give you. Domains need to be

bought and renewed year-on-year and can vary wildly in price. The companies that trade in domains tend to be a bit sneaky and will also often try and sell you stuff on top of it. Moniker are meant to be OK, and generally I recommend picking one company and sticking with it.











The steps for pointing your domain name to your host depend on where you're hosting your site! For services like Github Pages, Netlify and Vercel, you normally need to point your domain at their 'load balancing IP'. You can do this by creating what's called an Apex or A record, or changing the one that already exists for your site.

For some reason, it can often be hard to find the servers to point your domains to -- I think this is because they sometimes change and there's a few different ways to do it, but for the sake of simplicity:

- **Github Pages:** 185.199.108.153
- **Netlify:** 75.2.60.5
- **Vercel:** 76.76.21.21

Free domains forever with one neat trick

I love making subdomains! This is a great way to host loads of different sites from a single domain. You create a new Apex record for each domain and point it to the IP address you want. It took me years to realise I could do this!!

	Type ?	Name ?	Data ?	TTL ?	Delete	Edit
<input type="checkbox"/>	A	@	75.2.60.5	600 seconds		
<input type="checkbox"/>	A	factory	178.128.37.165	1/2 Hour		
<input type="checkbox"/>	A	library	134.209.77.44	600 seconds		
<input type="checkbox"/>	A	tcm	188.166.72.115	1/2 Hour		
<input type="checkbox"/>	A	transparencia	46.101.36.72	1/2 Hour		

You can tell from this screenshot that my website is hosted on Netlify :)

SSL

These days, all browsers will by default use something called 'SSL', the 'secure sockets layer', which is used to encrypt web traffic. This is done to stop malicious sites from being able to e.g. steal your card details. In order to work over SSL, you need something called an SSL certificate -- services like Netlify, Pages and Vercel will issue one automatically, and we manage the certificates for the CCI Server.

If you are configuring your own server, however, you will need to do this yourself! This is made much easier using a service called LetsEncrypt: Digital Ocean have a good guide to this [here](#).

3. Practical: making a static site and putting it online

For this part of the workshop, we will use a template to make a website, but we will also go through the process of setting up from blank files. You can download the template [here](#).

Download and unzip the folder, and then right click to open the 'index.html' file in your browser.

Modifying your site

The template contains basic HTML, CSS and Javascript. The only thing you actually *need* to have a website is HTML. CSS allows you to change the *look* of the website, and Javascript is used for interactivity.

First of all, have a go at editing the html file. This will let us change things like text and content on the page. For this workshop, we're making small informational websites about a favourite fact, artwork or object. W3 Schools has an intro guide to structuring HTML elements [here](#), and a full reference for all the different kinds of elements [here](#).

We'll also look at using [CSS](#) to style the site. Because it's a brief workshop we're just going to cover the basics!

Putting the site online

To put this site online, we are going to use Github Pages, following [these steps](#). You could also use netlify, vercel or another static hosting service.

bonus round: making a sticker

For the last part of the workshop, we're going to encode the website urls in NFC stickers. There's a great [free and open source phone app](#) you can use to do this, called NFC tools. This is inspired by a project from the web designer [Spencer Chang](#).

:)

