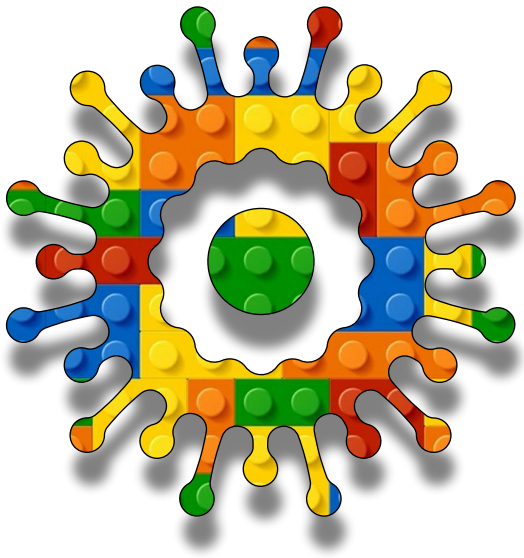
Algorithmic Intelligence Module A Unit #2 introduction to p5.js



Content

Module A Unit #2 introduction to p5.js

What machine do I need?
The editor
Signing in
The default sketch
Which format?
Looking at the buttons
The run and stop buttons
Naming the sketch
The auto refresh
Making the most of the general settings
Being more accessible
The main menu headings
The File heading
The Edit heading
The Sketch heading
The Help heading
The English heading
The purpose of the console
Accessing the sketch files
Adding files
The index.html file
Uploading files
Creating folders

What machine do I need?

This is probably the first question you might ask: Which machine should you use or not use for this tutorial? The simple answer is anything with a web browser will work for nearly everything. I do all mine on an iPad, but you can do it on a Chromebook, PC, Mac, laptop, smartphone, tablet, or a Raspberry Pi.

What you will need later on is a webcam and microphone; ones that are built-in will be fine; otherwise, you will need to connect your machine of choice to them.

Some issues I have experienced

On the iPad, I could not download the model; not sure why; it did on the Chromebook, and I assume it does on everything else, but if you are using a tablet or a smartphone, I cannot guarantee it. It is only one unit and is not critical, but that is the only issue I can find.

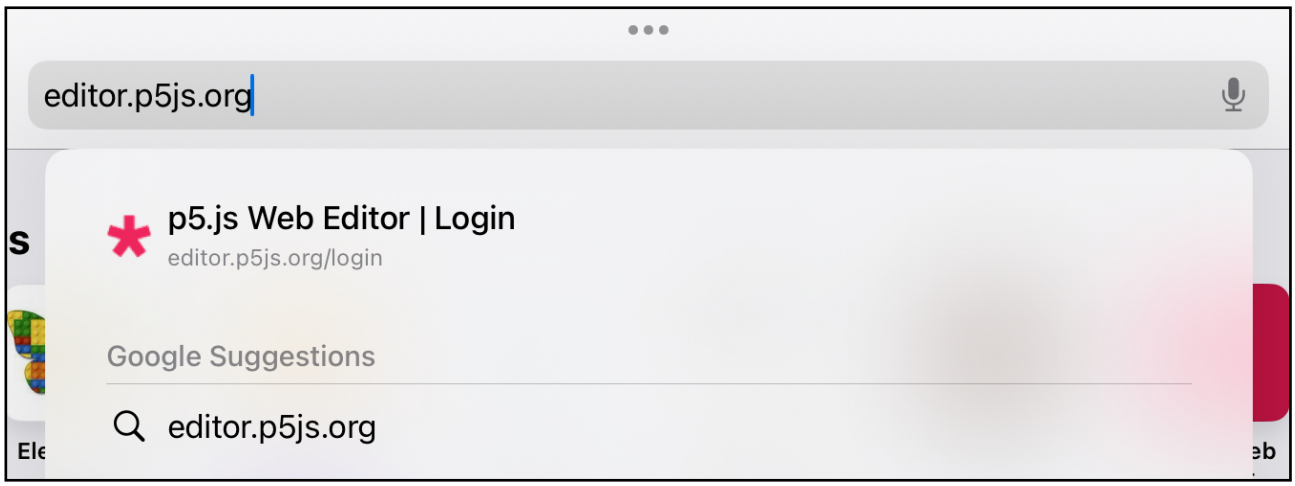
The editor

You don't need to download any software; you simply type this:

`editor.p5js.org`

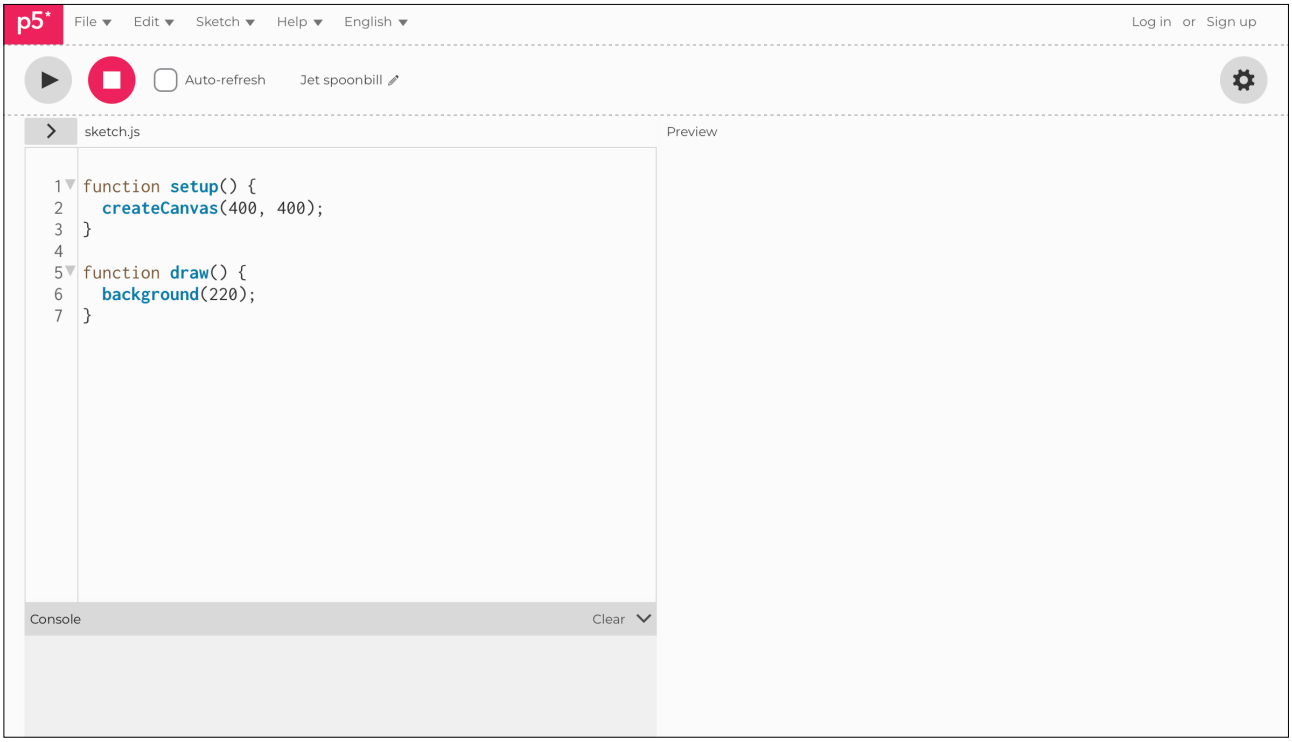
into your web browser and off you go. Most web browsers support it, but if you're unsure, then use Chrome or Safari. If you have a favourite one, then use that and see how you get on.

Figure 1: Type editor.p5js.org



When you first start, you get a page like this (shown below). You have a main menu across the top left-hand side. On the right-hand side, you have `Sign up` and `Log in` tabs. Below that, you have a number of buttons, a checkbox, and a text box.

Figure 2: The editor



Below that, you have two main panels. In the first panel (left-hand side), there is a column of numbers which are the line numbers for your code. The second panel is where you type your code. The third panel is the canvas, where stuff happens; the canvas will appear when you start to run your code.

If you press the run button (dark grey triangle in a light grey circle), the canvas should appear as a grey square where it says `Preview`. Underneath the panel for your code is another one called `console`; this is where your error messages and other information you request (in your code) will be sent and seen.

I will go through all these in a little bit more detail shortly, but for now, click on everything and anything to get a feel for the buttons and tabs. Everything is pretty intuitive, and the best thing to do is learn through playing with the buttons and seeing what happens.

Signing in (or logging in)

Although you don't have to sign up for anything and you can code straight away, you won't be able to save anything unless you have an account. I would seriously recommend setting one up right from the start. They have never, ever pestered me; it just used to log in.

Use one of the methods shown below and you are good to go (again). You can log in with your Gmail or sign up with your own email address. Test it all out first to make sure it works. I use `Login with Google` (recommended), but you can always use one of the other methods. Obviously, you will need a Gmail account for that. Sign up first and then log in.

Figure 3: Signing Up

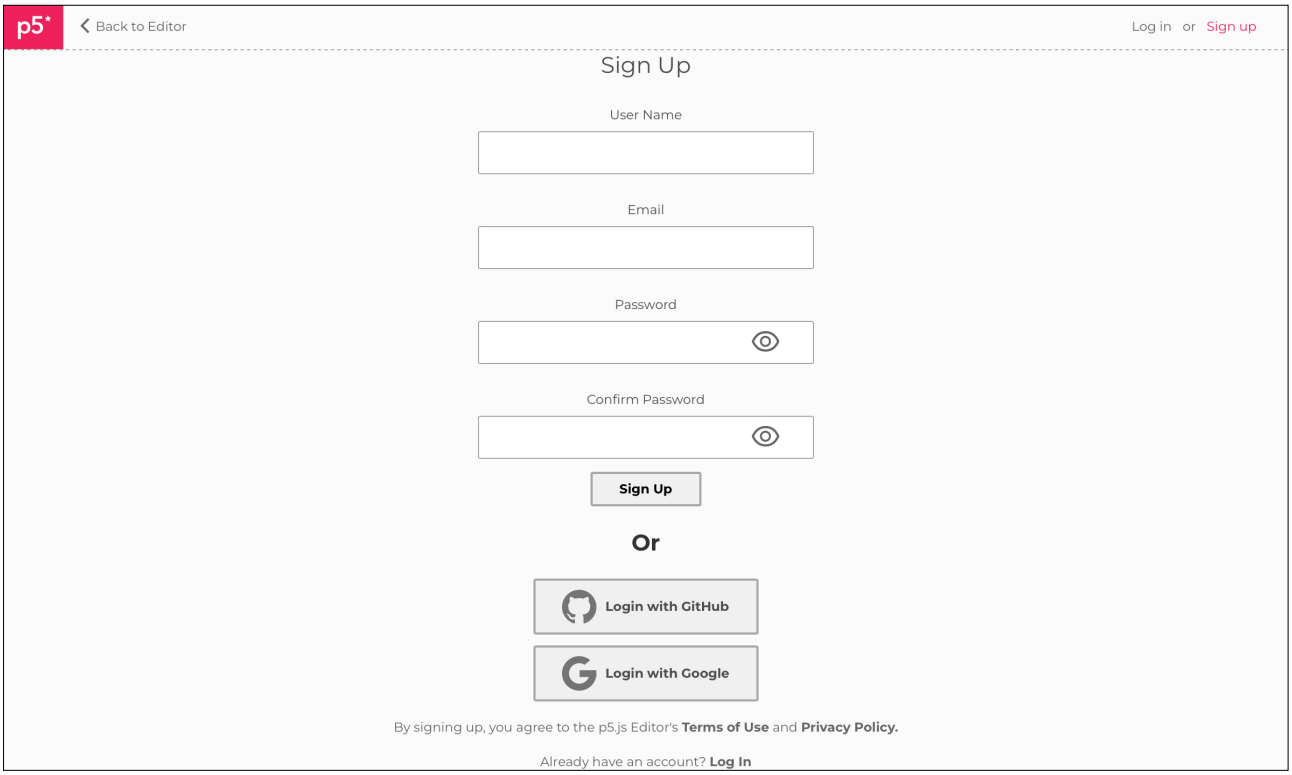
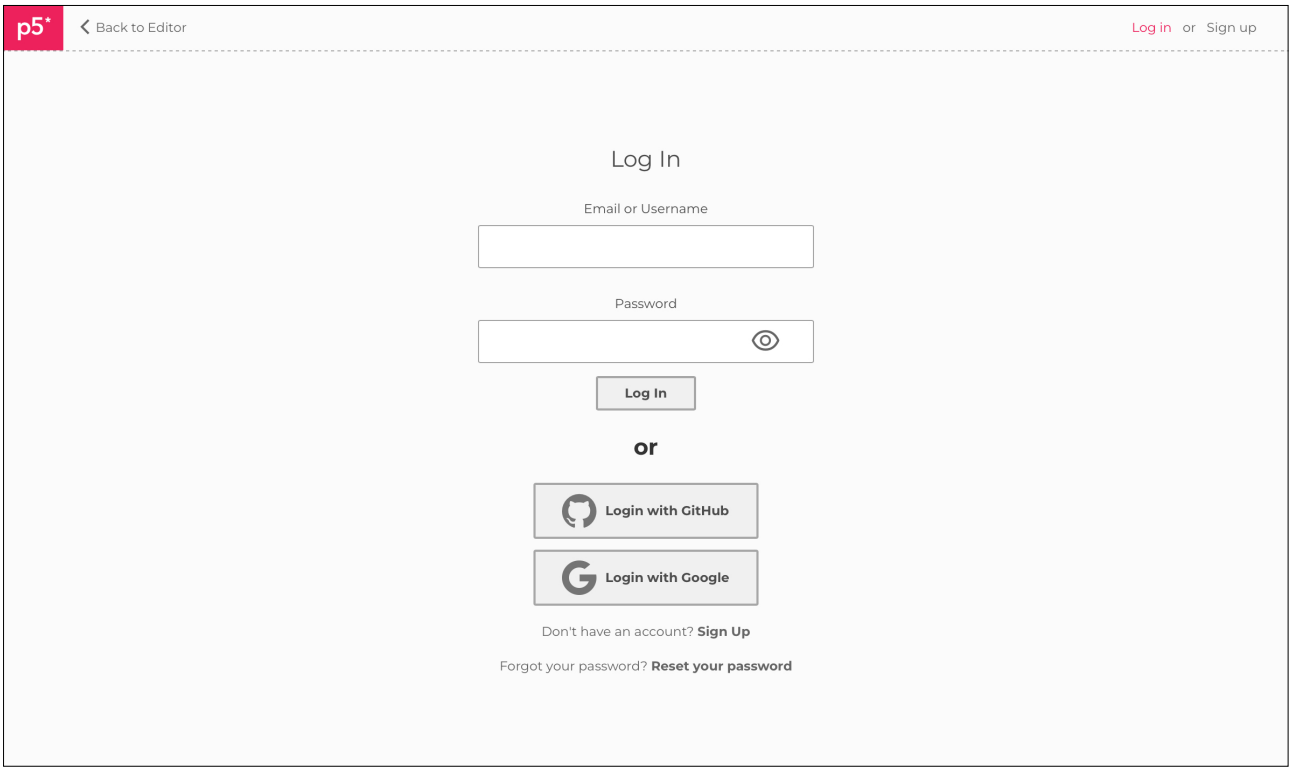


Figure 4: Logging in



The default sketch

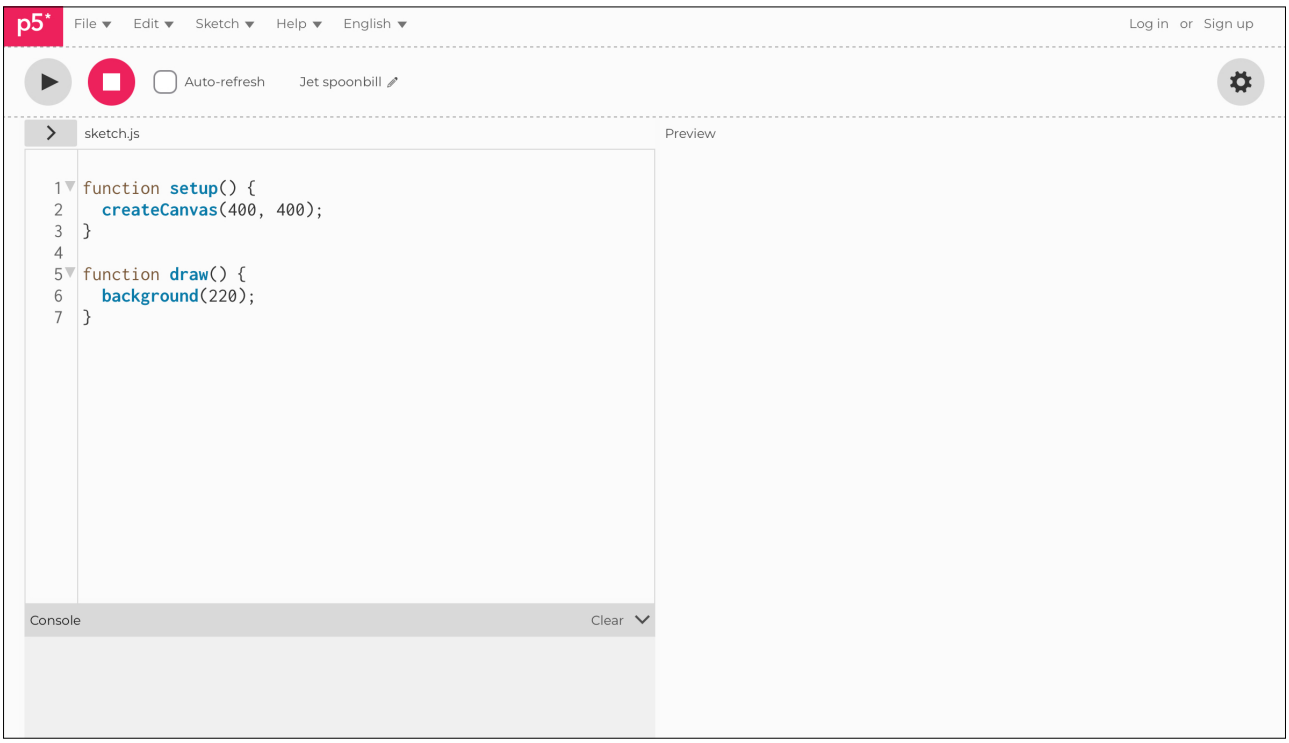
At the very start, you get a default sketch. This includes two functions and some curly brackets. These two functions are important and are built-in functions. The first one, `function setup()`, is to set up how you are going to see and use the sketch; this runs through once only and usually includes the size of the canvas you want.

The second one, `function draw()`, is a loop; it draws on the canvas continually. This is where much of your code will go, although in some cases, you won't even have a `draw()` function, but more on that much later.

Inside the curly brackets is where your code sits. At the moment, you have the canvas size of `400` pixels by `400` pixels in the `setup()` function. In the `draw()` function, we have a grey `background()` with a value of `220`. This is the grey value from `0` (black) to `255` (white) and everything in between. For most of my sketches, I will simply use this value (`220`) for simplicity.

You will also notice the semicolon (`;`) after some lines of code. Most people who code with p5.js include this at the end of lines of code. I have omitted it to keep the code cleaner and easier to read. It has never been a problem not to use the semicolon, but it is up to you to decide whether to keep it in. This is just my preference, and I am in the minority here.

Figure 5: Default sketch

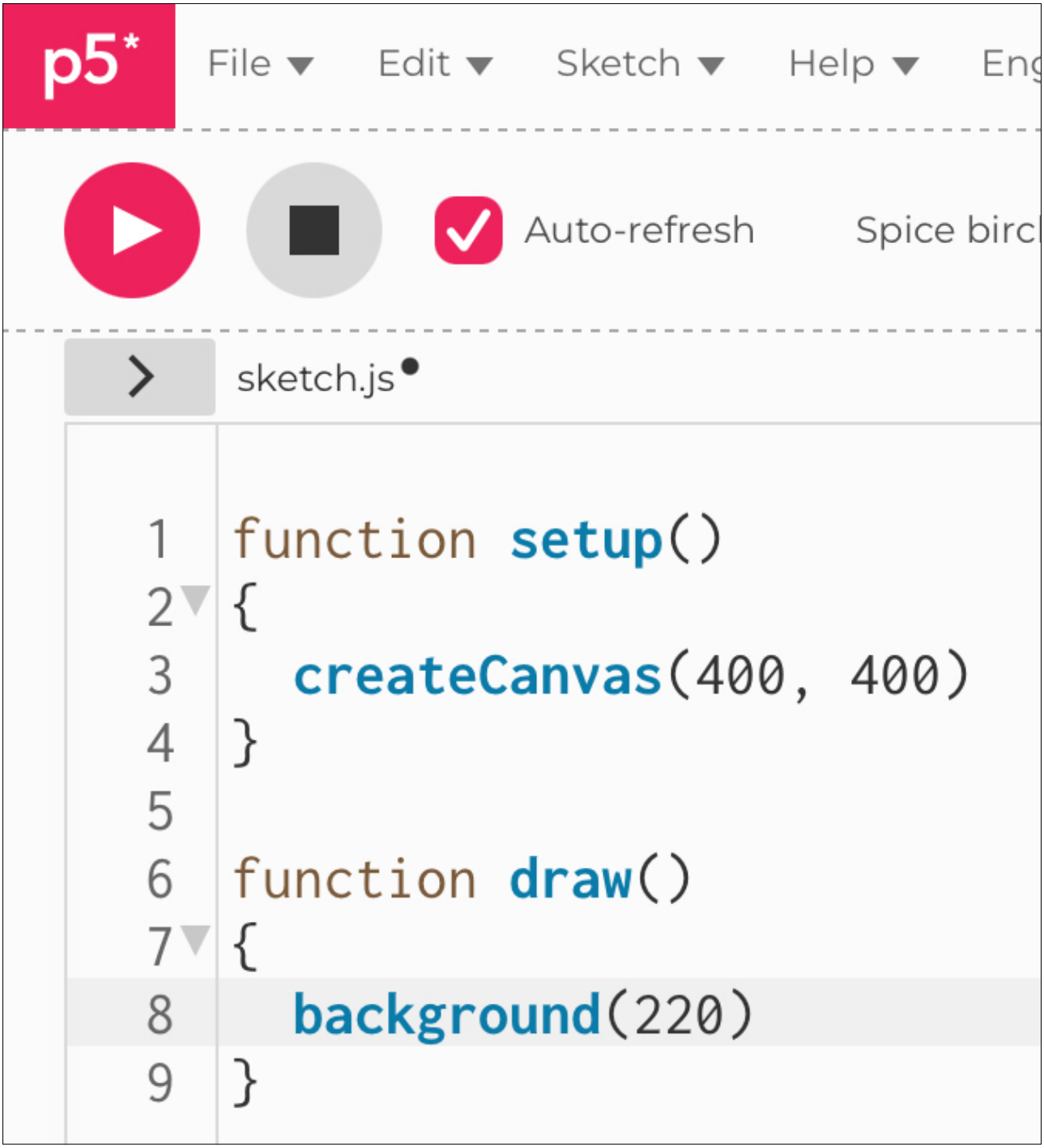


Which format?

There is the default format, but the format I will present in my tutorial will be different. You will see that the first curly bracket starts on the same line as the function name; it is how most people will use this, but I want to make the curly brackets { } very clear, and I change that to the first curly bracket is on the next (first) line after the name of the function. This is just my preference for clarity.

The lines of code are placed inside a set of curly brackets { }; there are always two, like bookends. The code inside a set of curly brackets is indented with two spaces; this is the default setting and, although not essential, makes the code readable and stands out clearly.

Figure 6: My format



Looking at the buttons

You will see a row of buttons across the top. It is worth spending the time getting to know them well. The first thing to consider is how to save your work as you go along. If you are working on something one day and want to continue the next day, then you need to save your work. Also, occasionally your code may crash, for whatever reason. This is why you need to save your work as you go along, not only at the end of a session. You can set it up to save your work automatically as you go along, but be aware there is no undo option.

It is worth exploring these buttons and also the tabs to familiarise yourself with what they look like and what they offer before you go very far into your coding. I will highlight the most important ones, but they all have their function and purpose.

Figure 7: Buttons

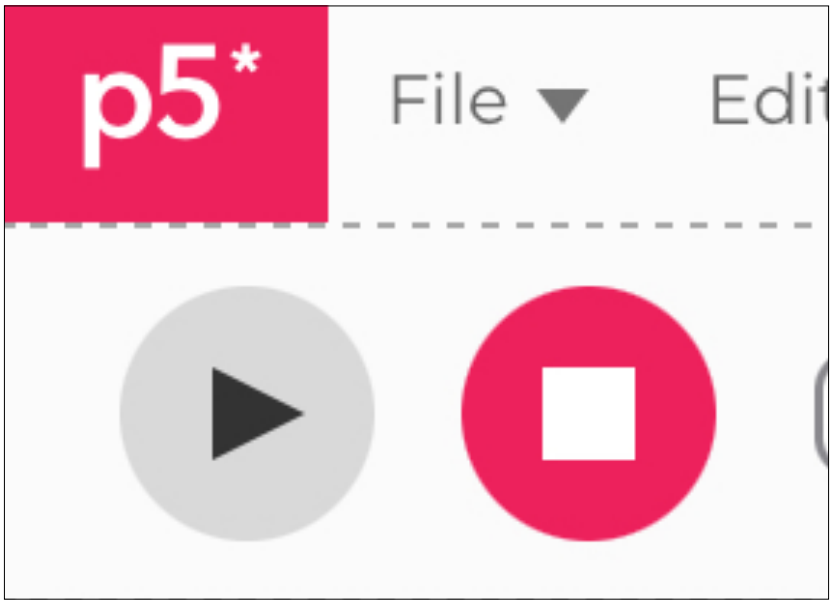
The run and stop buttons

The first button you come across is the run button. It looks like the play button on a device you might have seen. After you have written your code, you will want to run the code to see what happens. When you click on the run button, it will do that.

To stop the code running, you will need to press the stop button, which is next to it with a square in a circle, again pretty intuitive. The programme will keep on running the code until you stop it.

So every time you write some new code, remember to press the run button each time. You can also set it up to run automatically every time you add new code, but I wouldn't recommend it until you really know what you are doing; this is because it can cause problems as it tries to run incomplete code, at which point you will lose any changes since your last save.

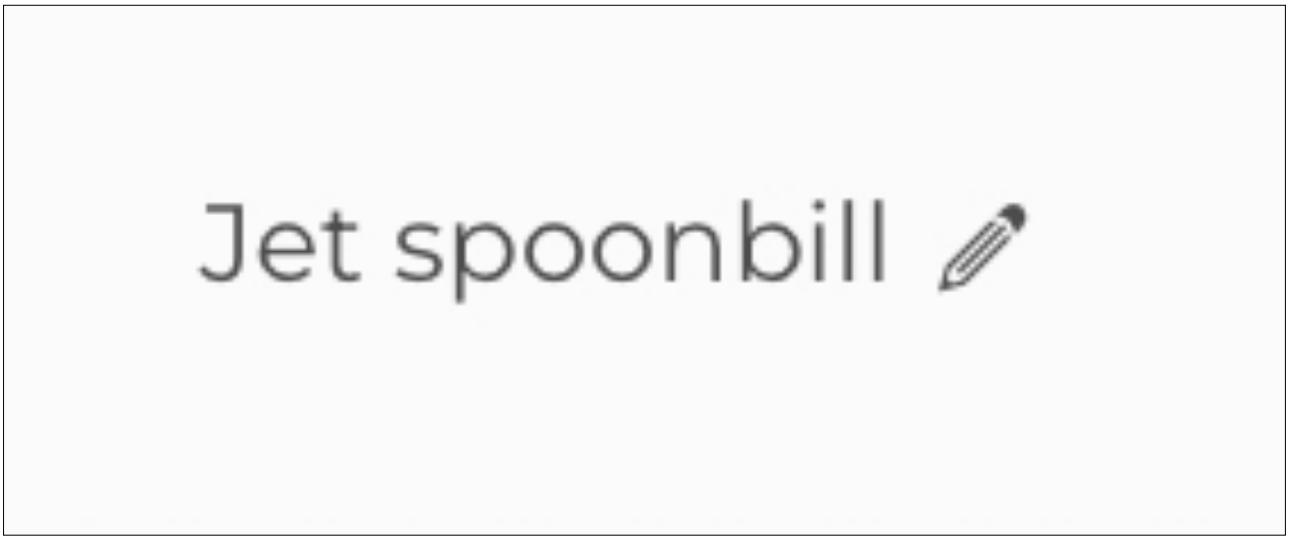
Figure 8: Run & Stop



Naming the sketch

You will have seen a box with a couple of random words in it. This is a randomly generated name for your sketch; in my example below, it generated `Jet Spoonbill`. There is a little pencil symbol next to the words. This means you can edit the name to something more meaningful or just leave it as it is if you wish. This is entirely up to you; a new set of words is generated each time you start a new sketch. This is another reason for creating an account so that you can save your sketch under that name and access it again later on, carrying on from where you left off.

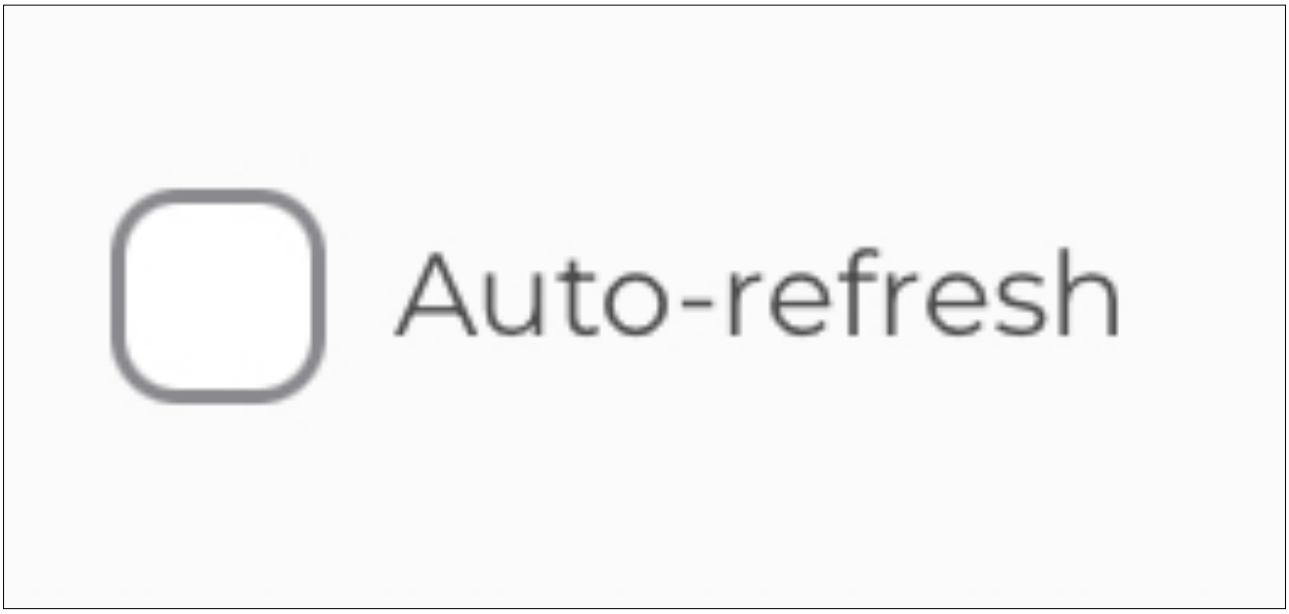
Figure 9: naming sketches



The auto refresh

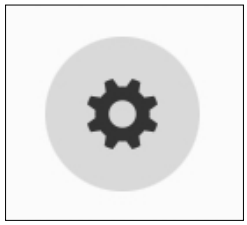
The button before the sketch name box is the auto-refresh button. If that is highlighted or selected, then I recommend that you unselect it by clicking on it. It may be unselected by default, in which case for now leave it as is, but later on when you are more confident in what you are doing, you may like to select that option.

Figure 10: auto refresh



Making the most of the general settings

Here we have a button on the far right-hand side which looks like a cog wheel (see opposite). If you click on it, you get a `settings` menu which is split into two parts. The first one is `General Settings` and the second is `Accessibility`.



In `General Settings`, you will have a list of options, most of which (if any) you don't have to alter:

Themes

Depending on what works best for you, choose a theme from a choice of three that is easiest on the eyes. I quite like the high contrast, but for this tutorial, I use the default `light`.

Text Size

It is pretty obvious, but bear in mind that if you make it too big, you will get long lines of code wrapping themselves onto the next line, which I try not to do for this tutorial. Choose the size that works for you.

Auto Save

It might be worth keeping it on for now, but sometimes I want to go back to the last version I saved manually, and if it has saved a later version, it can be an issue, but it is swings and roundabouts whether to have it or not. My preference is to leave it off and to manually save the sketch after each significant change. This is a discipline that is easy to forget. Leave it on if unsure at this stage; just practice saving as you go along.

Autoclose Brackets and Quotes

This is a useful one to keep selected; you don't have to, but I would recommend it. You will see the difference if you switch it off; it saves you a bit of time as you will see.

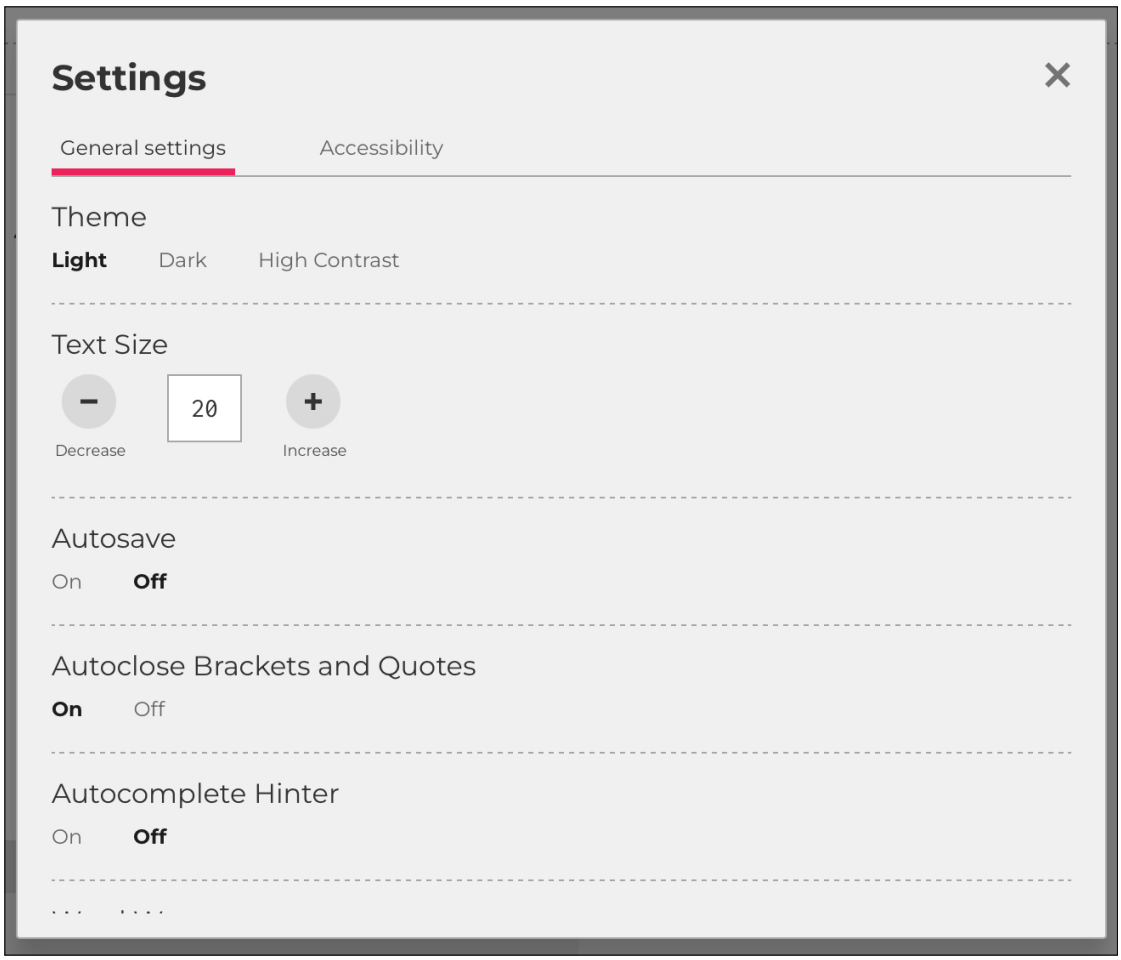
Autocomplete Hinter

I found this one very annoying if left on. I suspect some coders like it and use it a lot, but I found it got in the way of my coding as it tried to predict what I was about to write; my strong recommendation is to deselect it if it is on by default.

Text Wrap

This is only an issue if the line of code reaches the edge of the panel. Leave it on if you always want to see the code and not disappear.

Figure 11: general settings



Being more accessible

The second menu is `Accessibility`.

Line Numbers

You can switch the line numbers off or on if you wish. The line numbers are useful if there is an error message, and they can often tell you on which line of code the error message relates.

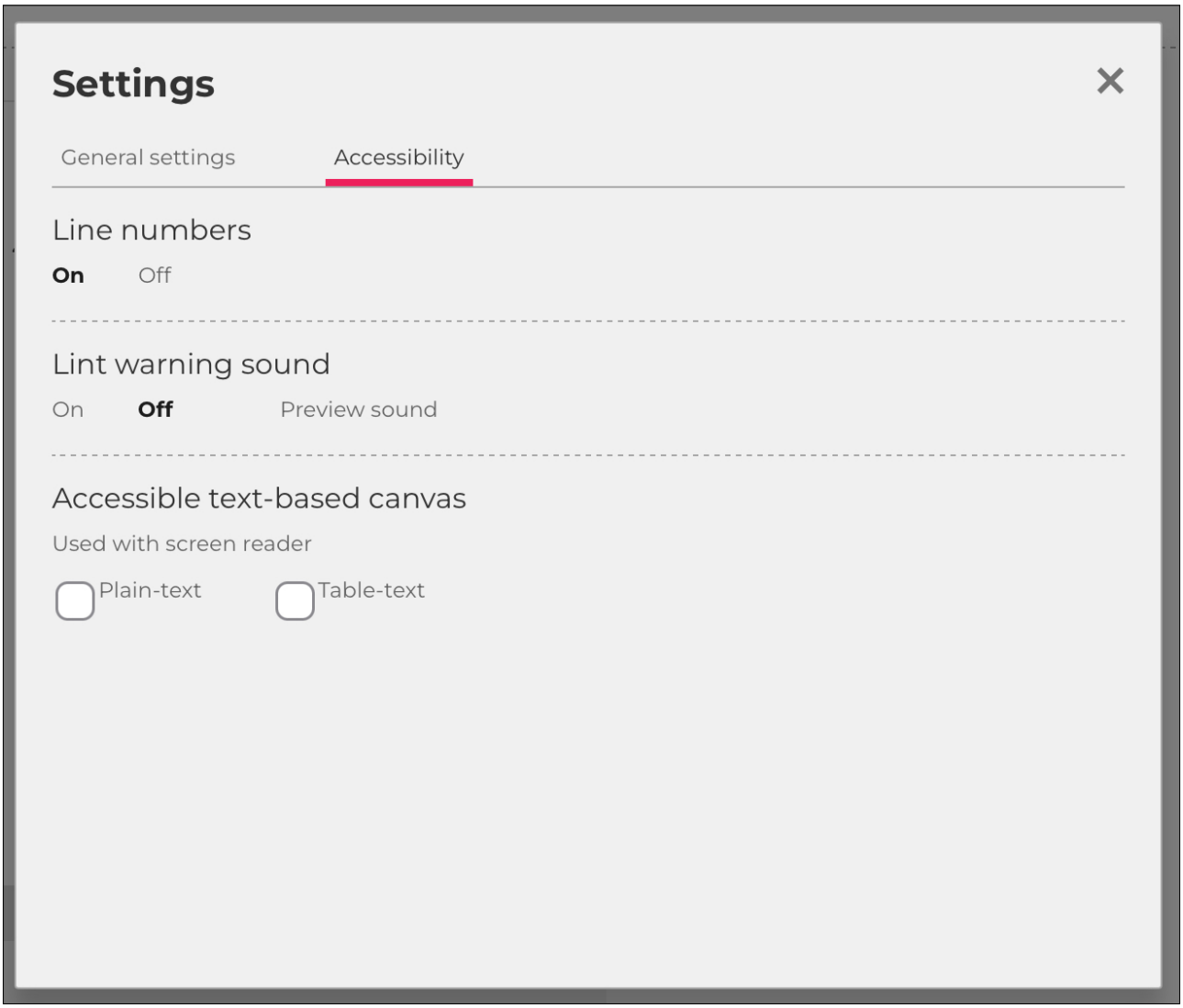
Lint Warning Sound

This gives you an audible sound if it detects an error in the code. Again, it is a preference to have or not have it selected. I don't; I think it would drive me mad after a while.

Accessible text-based canvas

Not entirely sure what these do, to be honest, but they may mean something to you.

Figure 12: accessibility



The main menu tabs

We have a list of main menu tabs or headings. If you click on any of the following: File, Edit, Sketch, Help, and English, you will get a subheading list. I will make reference to some of the key subheadings that you are likely to use.

The main heading you will use is File as it is for managing your sketches. The Edit is also useful for finding words and/or replacing them. If you tidy your code, it will do just that and it will do it in the default format with the curly brackets and the semicolons. The Sketch is where you can add files, folders, run, and stop; there are other ways to do that which I will show you later (run and stop I have already covered). The Help is where you can get some additional information, and English is where you can change the language used.

Figure 13: main menu

The File menu

Under the File menu, you get either a short menu if you have not saved your sketch, or a longer menu if you have. I am showing you the longer version.

New

This gives you a brand new sketch with the default code and a new, randomly generated name.

Save

This is something in the web editor. Use it as often as you can, just in case something goes wrong unexpectedly, so you have a backup up to that point. I generally don't learn shortcuts, but this is one shortcut worth learning, as you might notice the command symbol plus the letter S. You press both together rather than going through the menu each time. There will be slight variations for different machines and keyboards.

Duplicate

You can find sketches on the internet or one of mine that you want to use. If you have found a sketch that you want to keep, use, or edit for whatever reason, then you can duplicate it (make a copy) and save it under a new name. That way, you can have your own version.

Share

You may want to create a link to your sketch or even embed it on your website. This tab gives you a number of options.

Download

This is something I have never used and am not sure when you would use it.

Open

This takes you to another part of the web editor where you will see a list of all your saved sketches. If you have a lot (and I do), then you can search for specific keywords. Another reason why it is important that you rename your sketch with something meaningful in case one day you want to find it. They are saved chronologically.

You have two other tabs under the Open tab: Collections and Assets:

Collections

This is where you can view any collections you have created, as well as creating them.

Assets

When you add images, videos, music, etc., this is where you can see them and which files they are linked to. There is a finite amount of memory space to keep them, but there is plenty for many sketches and numerous images and files.

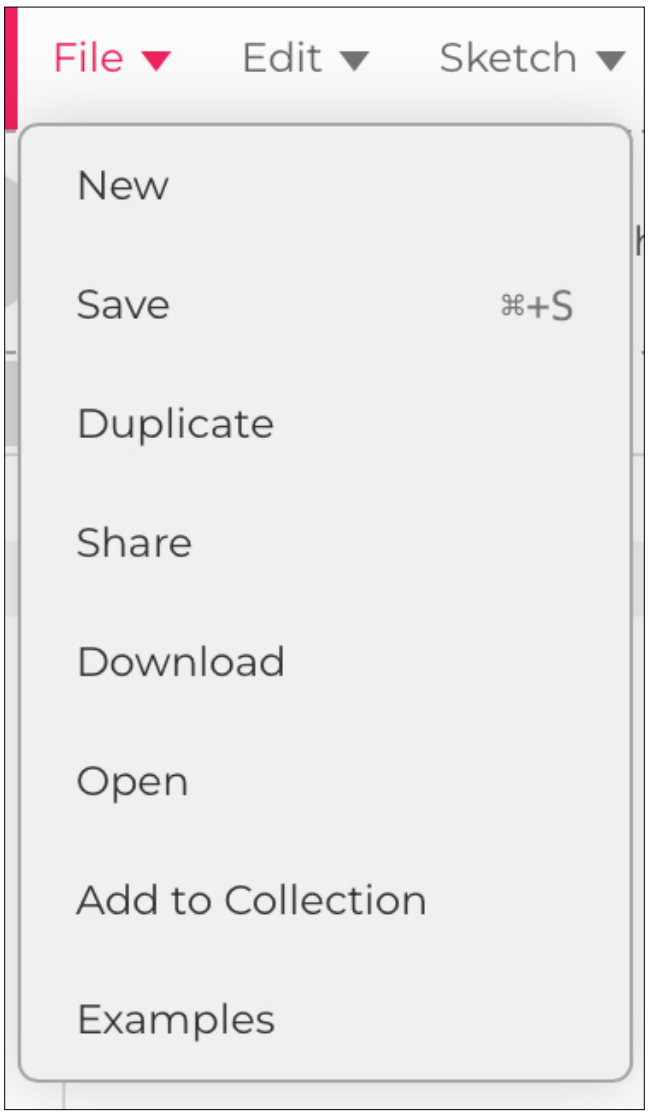
🀄 Add to Collection

One way to sort and manage your sketches is to make a collection and add the sketch to it. This is similar to a folder to hold a specific collection of sketches and can be useful if you start creating sketches for different purposes, for instance, games or art.

🀄 Examples

A large repository of useful examples, well worth a quick explore to see some interesting stuff.

Figure 13: file



The Edit menu

The Edit menu gives you some useful functions.

Tidy Code

This will tidy your code but will put all the semicolons back in and move the curly brackets to where they are normally kept, so it will change the appearance of your code considerably, assuming that you have adopted my format approach.

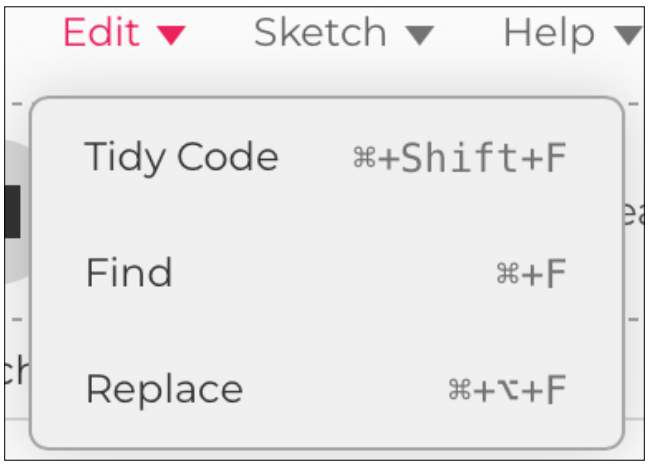
Find

This will search for any specific keyword or phrase in your code.

Replace

This gives you the option of finding a particular word or phrase and an option to replace it. You might use this to change the name of a variable.

Figure 14: edit



The Sketch menu

The Sketch menu offers an alternative place to run a number of actions.

Add File

This is where you can create another JavaScript file, such as vehicle.js. You can also do this from the sidebar.

Add Folder

This is where you can create a folder to put a collection of files or upload images, videos, models, or data files. You can also do this from the sidebar.

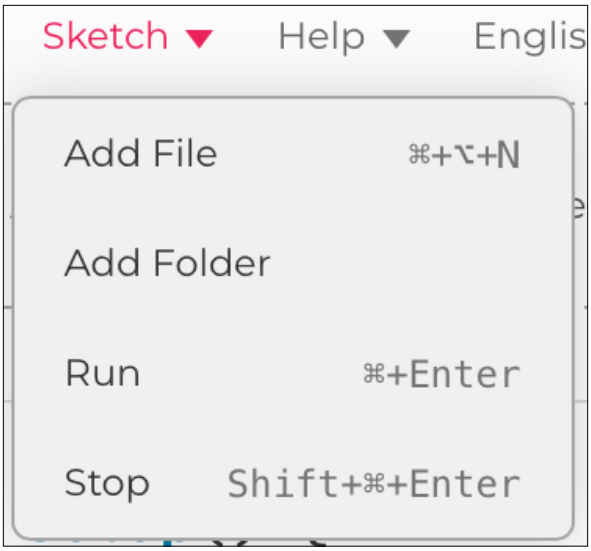
Run

This does the same as the button.

Stop

This also does the same as the button.

Figure 15: sketch



The Help menu

The Help menu is quite useful and worth a look at.

Keyboard Shortcuts

There are quite a few, and some are worth learning, such as Save, but others are probably not so useful; it all depends on whether you are generally good at learning shortcuts and would use them.

Reference

This opens up a new tab in your browser. This is an immense resource and can look a bit bewildering. There is a search box, and there are lots of examples and a reference section. It would take too long to go through it, but I would highly recommend having a look at it.

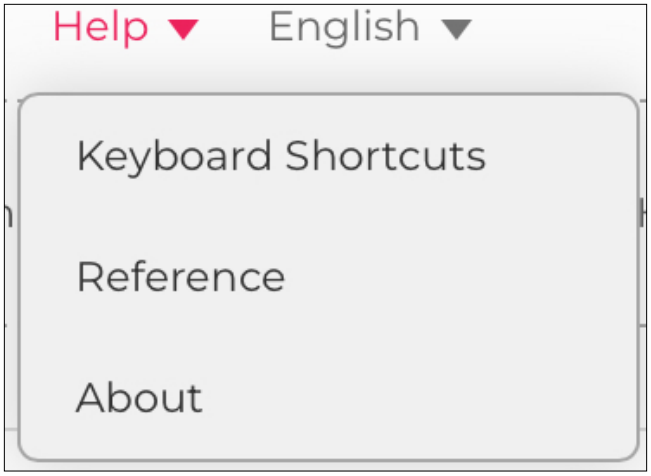
About

This has some links and general information.

English tab

You can change the language to one of many if your first language isn't English.

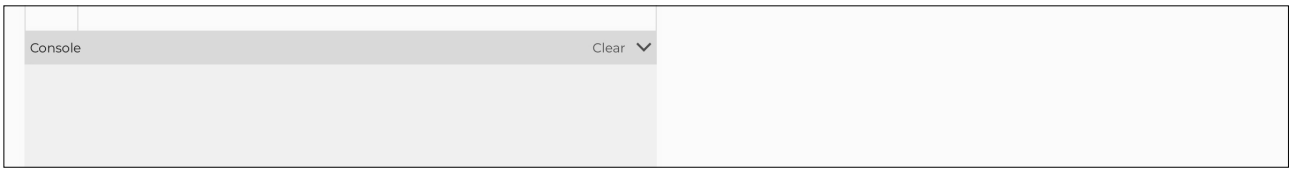
Figure 16: help



The purpose of the console

The `console` is the grey-boxed section underneath where you enter your code. This is a very useful function for a number of reasons. It is where you will get error messages and where you can send information. Its main purpose is for debugging, which is another word for problem-solving.

Figure 17: the console



Error messages

If your editor picks up on some glaring and obvious errors in your code, which can be anything from a missed comma or semicolon to an unknown variable appearing. Mostly you will get something useful and informative, although sometimes it just flags a problem and leaves you to search for it.

Sometimes the error code may be highlighted in red, and it may even give you the line number to identify where the problem is (or the following line number where it does become a problem).

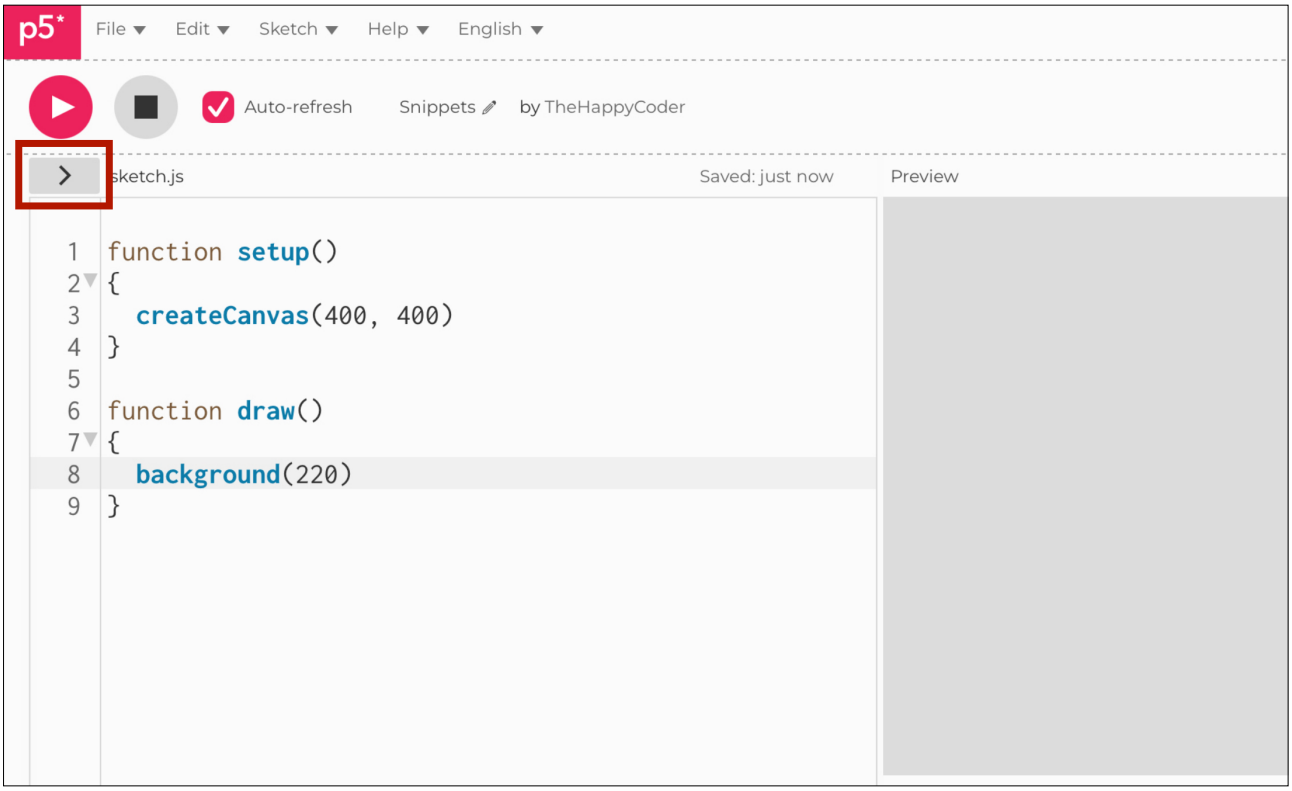
Console log

We can put a line of code in called `console.log()`. This means you are going to log something in the console. Sometimes it can be a piece of text such as `console.log('finished training')`, or it can be the value of a variable such as `console.log(counter)`, which will give you the value of the variable at that time in the code. Another really helpful use is to debug a problem where you are not sure what is happening, for example, to an array.

Accessing the sketch files

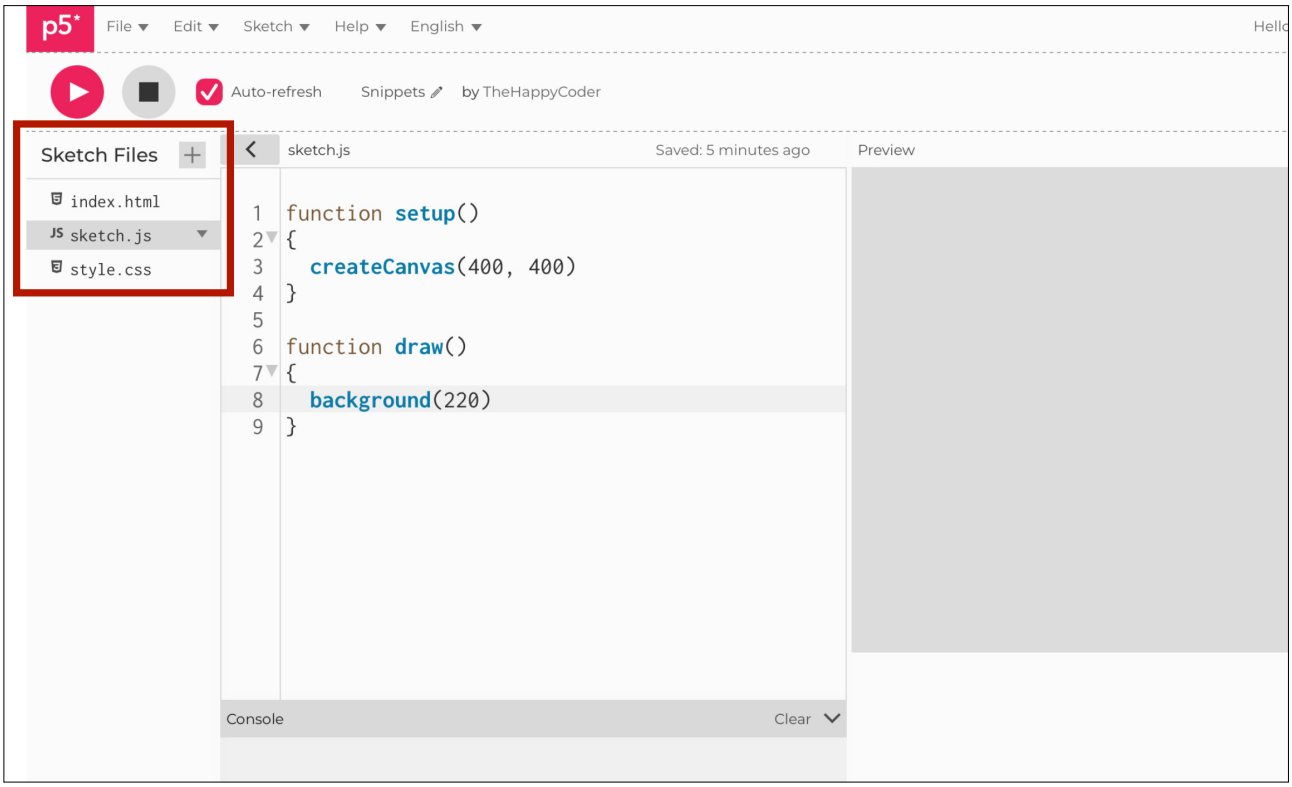
It is going to be critical to understand where the sketch files are and what files are used in creating a sketch. To the left of where you type your code, you will find a grey box with a grey arrow. I have highlighted it in red; click on it.

Figure 18: sketch files



When you click on it, you will get a menu of `Sketch Files` associated with the editor. Those listed here are `index.html`, `sketch.js`, and `style.css`. The main ones you will be interested in are `sketch.js` and `index.html`. The style.css can be ignored (for now).

Figure 19: files menu

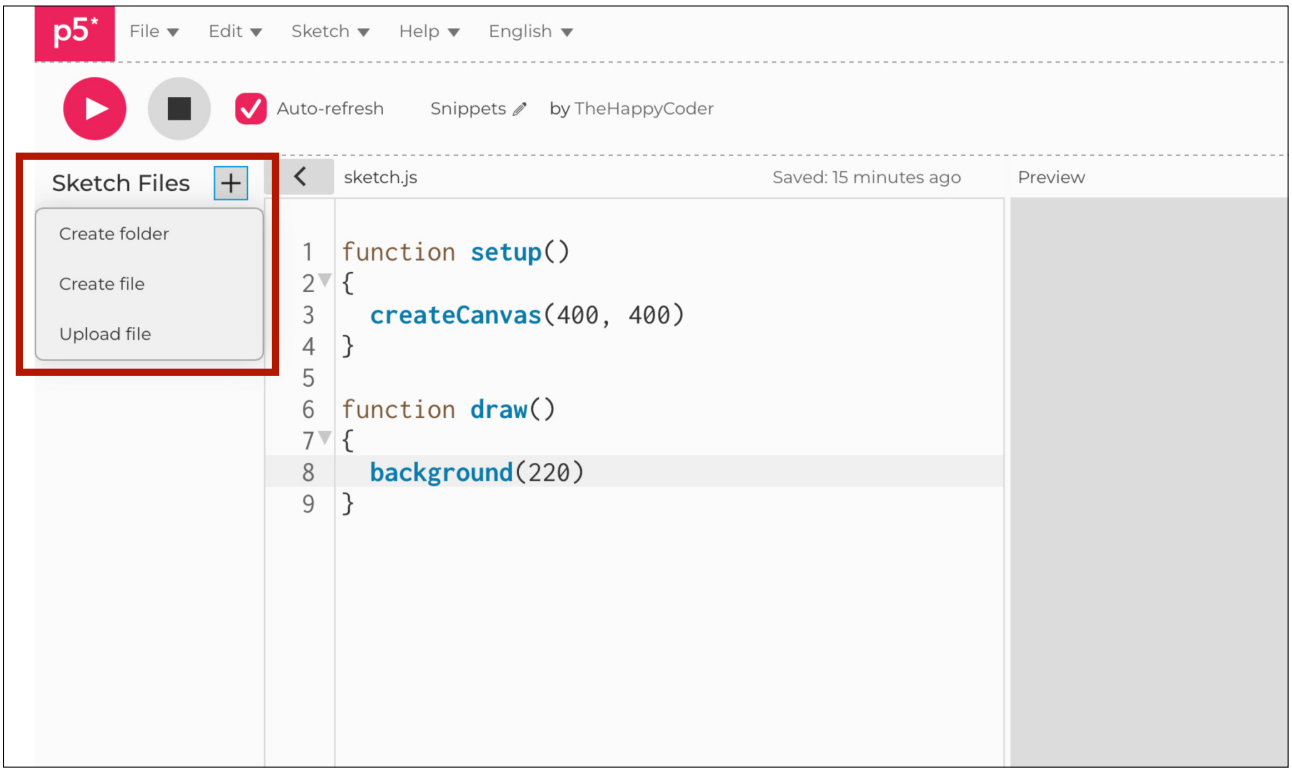


Adding files

Although I have avoided adding extra files in the majority of the coding, there are times when adding them makes things neater and, on the whole, easier to manage. You have the `sketch.js` file by default as your main coding file, but you may want to add more. You will need to give it a name and file extension `.js`, for example, `particles.js`.

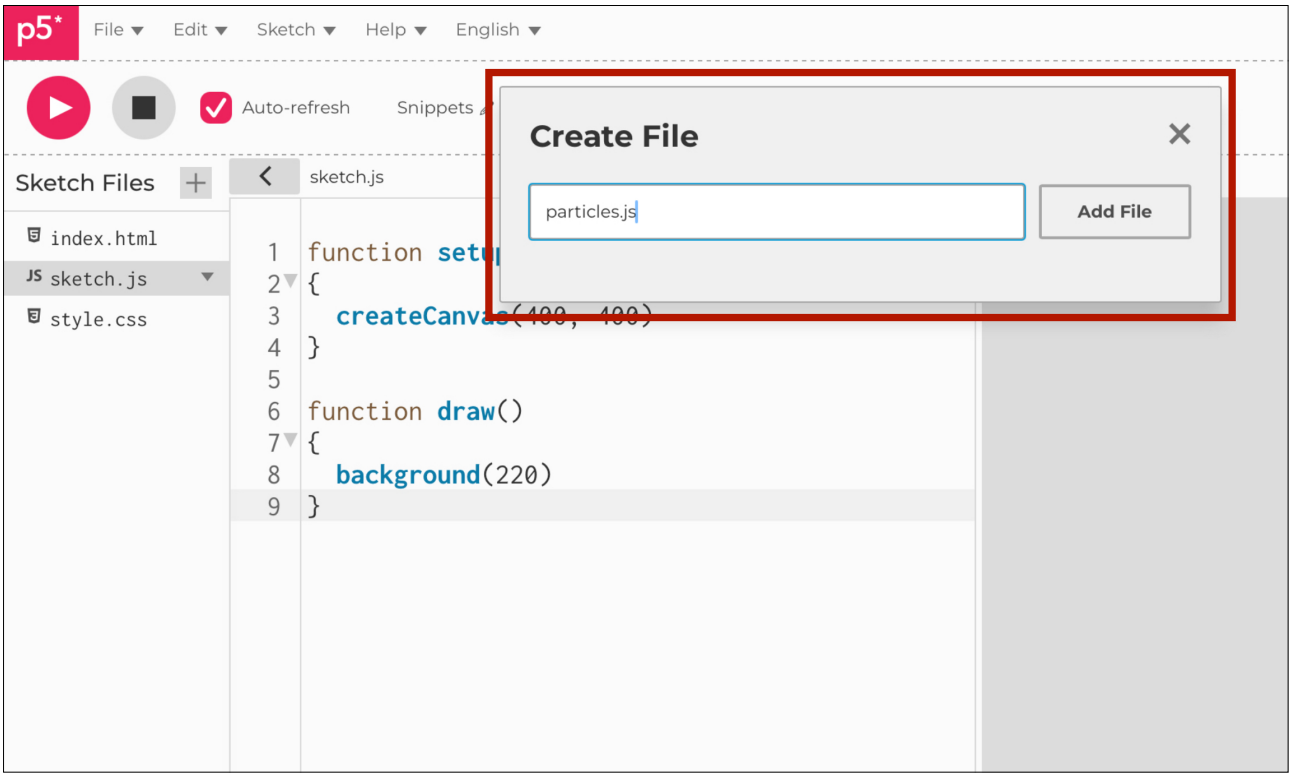
1 You add them by clicking on the grey `+` sign next to `Sketch File`.

Figure 20: adding files



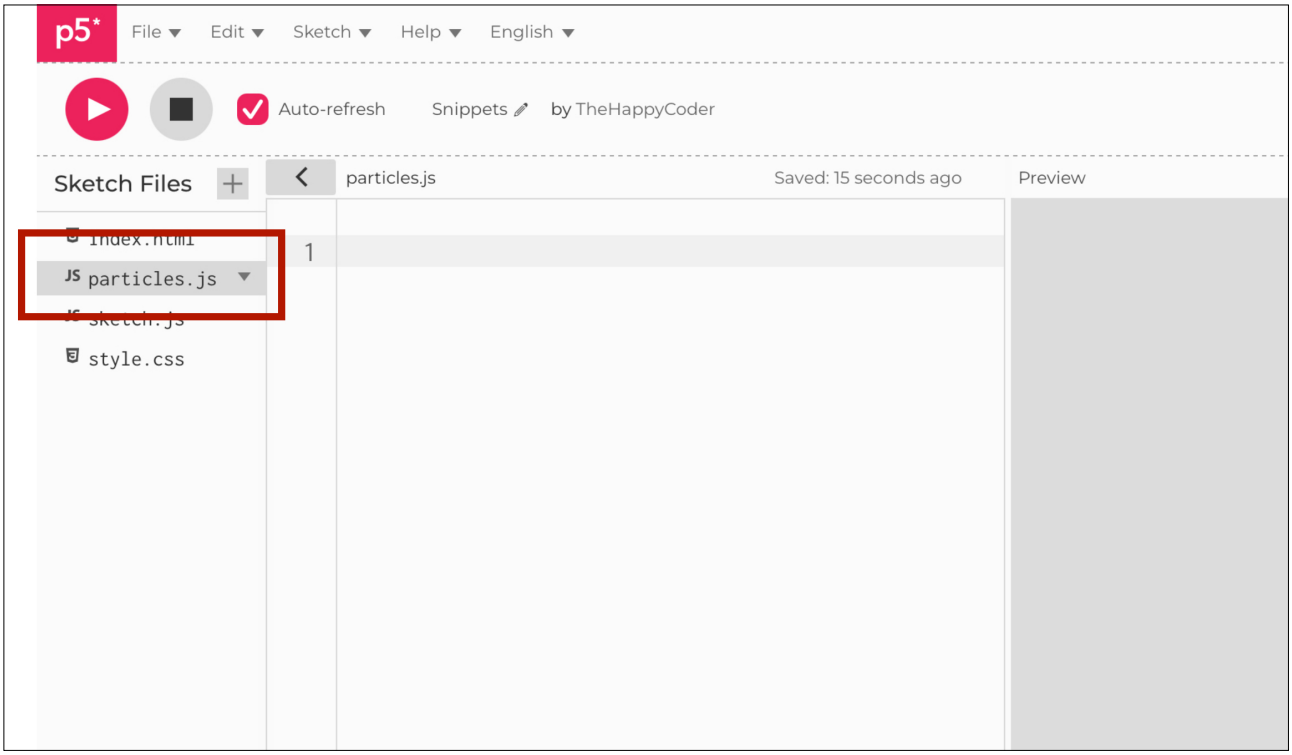
2️⃣ Now you click on `Create File`. You will get a box appearing waiting for you to put the name of the file in (it is important that it is case-sensitive and must have the `.js` file extension at the end).

Figure 21: creating and naming files



3️⃣ Click on `Add File`, and the file will appear in the `Sketch files`. Notice that the file is empty of any code. However, we aren't done yet; to use this new file, we have to make reference to it in the `index.html` file.

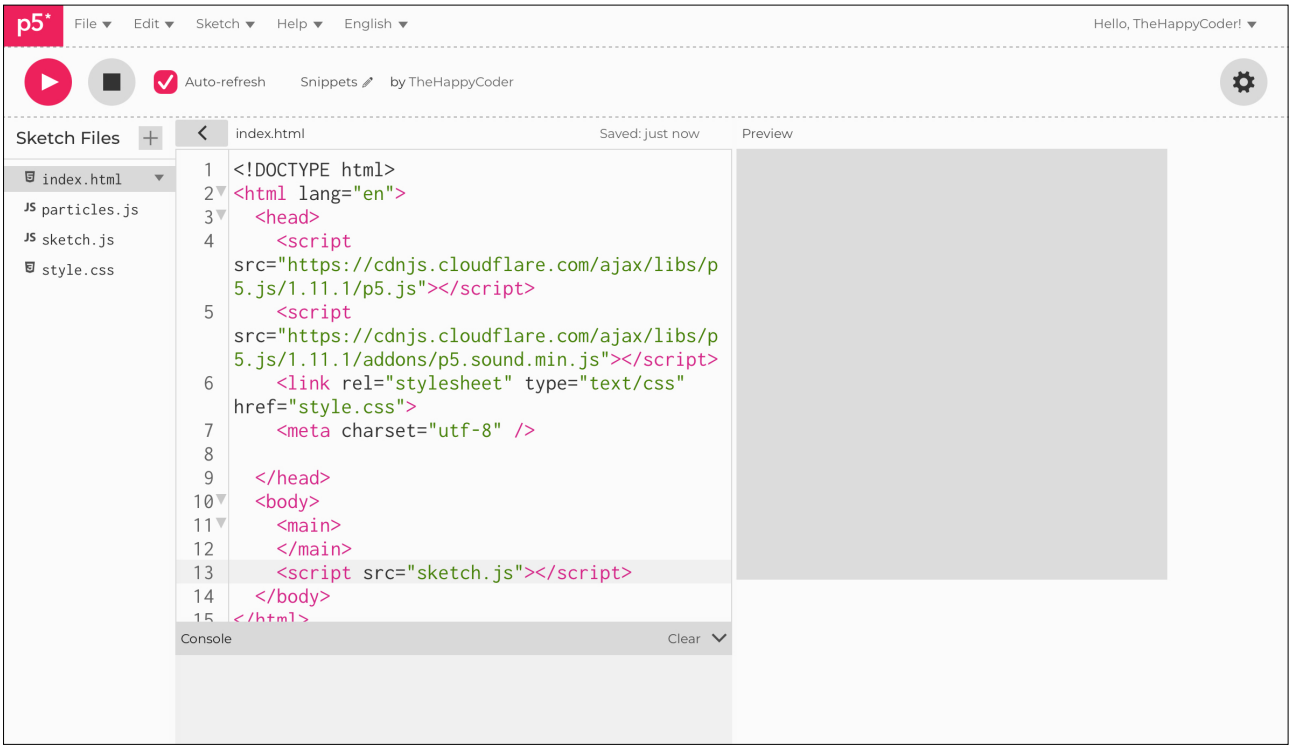
Figure 22: adding the file



It is worth showing you now because I will make reference to it later when adding the `ml5.js` library (and other files), but for now I will show you how to add the file we have just created. This bit is often easy to forget to do and then wonder why the code is throwing a wobbler at you.

4 This time we click on the `index.html` file to open a bewildering amount of code. Don't be put off if you are not familiar with HTML tags.

Figure 23: the index.html



This is the code within the index.html file. The index file is the core of the editor because it is where the editor interacts with the web browser. HTML is a tag-based coding language and is often used to create websites. All this is in the web editor by default. We don't need to install the p5.js code; instead, we access the library through a link. Here is a brief explanation:

```
<script  src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/2.1.1/p5.js"></
script>
```

The latest version is 2.1.1 (as of writing) but the default is 1.11.1 until late 2026. You can use either but there some things that won't work in version 2, that worked with version 1. I am going to ensure that most things will work for both.

```
<script  src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.11.1/addons/
p5.sound.min.js"></script>
```

Although the library for ml5.js is not there by default (the other two are), we can add the ml5.js thus:

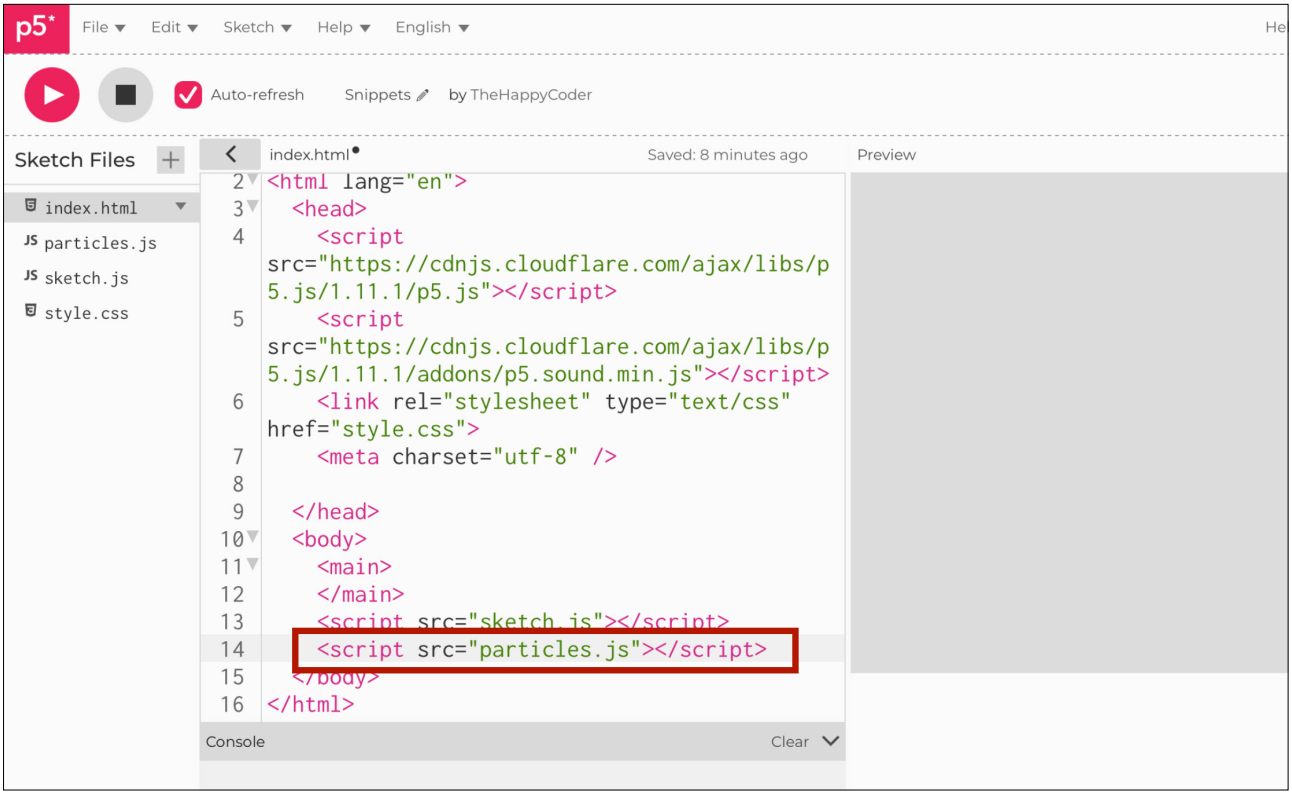
```
<script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></script>
```

This line of code needs to be added when you want to use ml5.js, but I will take you through that later and even offer you a link to a blank sketch rather than typing it all out. The current version is shown above; there is an older version which includes some examples, but be aware that the older sketches don't work with the newer version without a little bit of tweaking. We will be using the latest versions of p5.js and ml5.js.

Notice that these lines have `<.../>` symbols; these are called tags. There is a lot I could share about JavaScript, CSS, and HTML, how they work together to create websites, how they give text style and functionality, but that is a topic all on its own and not essential for this tutorial, although it might include one at some time in the future.

5️⃣ The key point here is the line of code that shows the `sketch.js` tags. What I do is copy and paste it underneath and change the second `sketch.js` to the new name of the file, in this case `particles.js`. It has to match the name exactly.

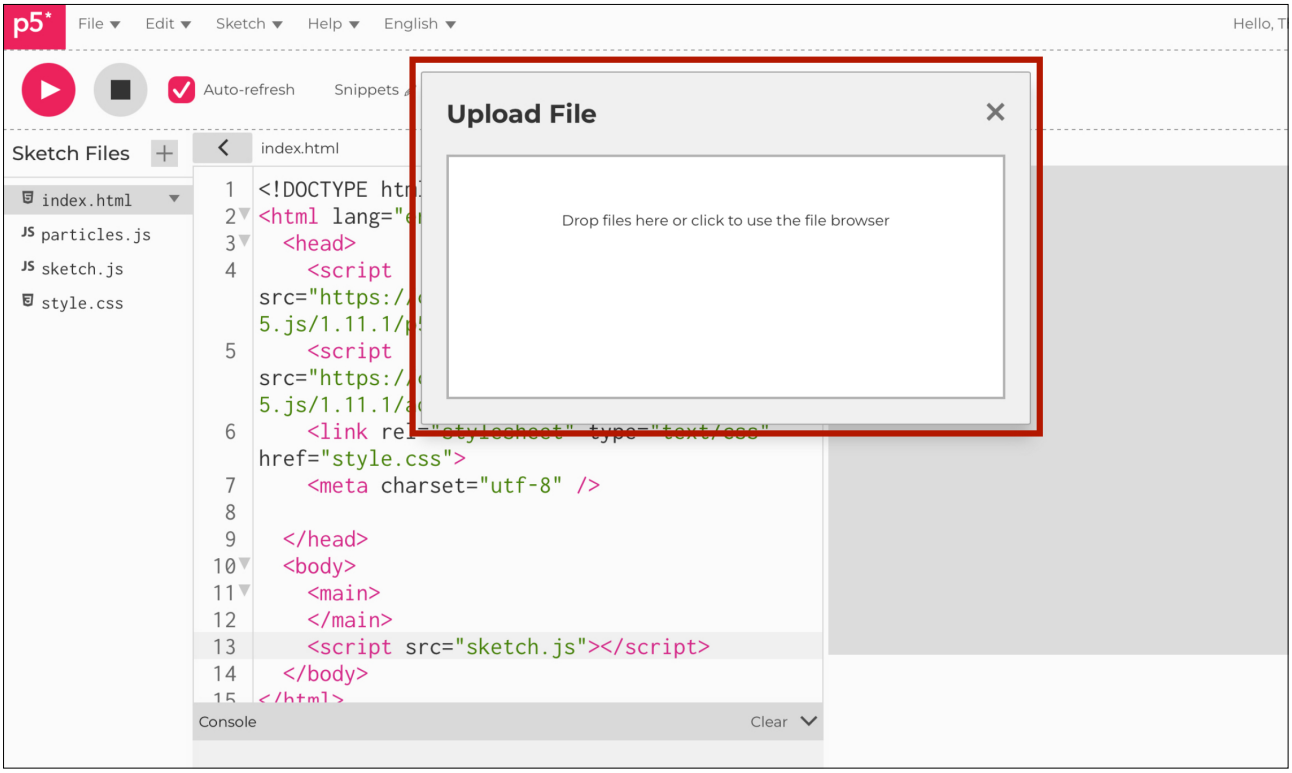
Figure 24: index.html new file



Uploading files

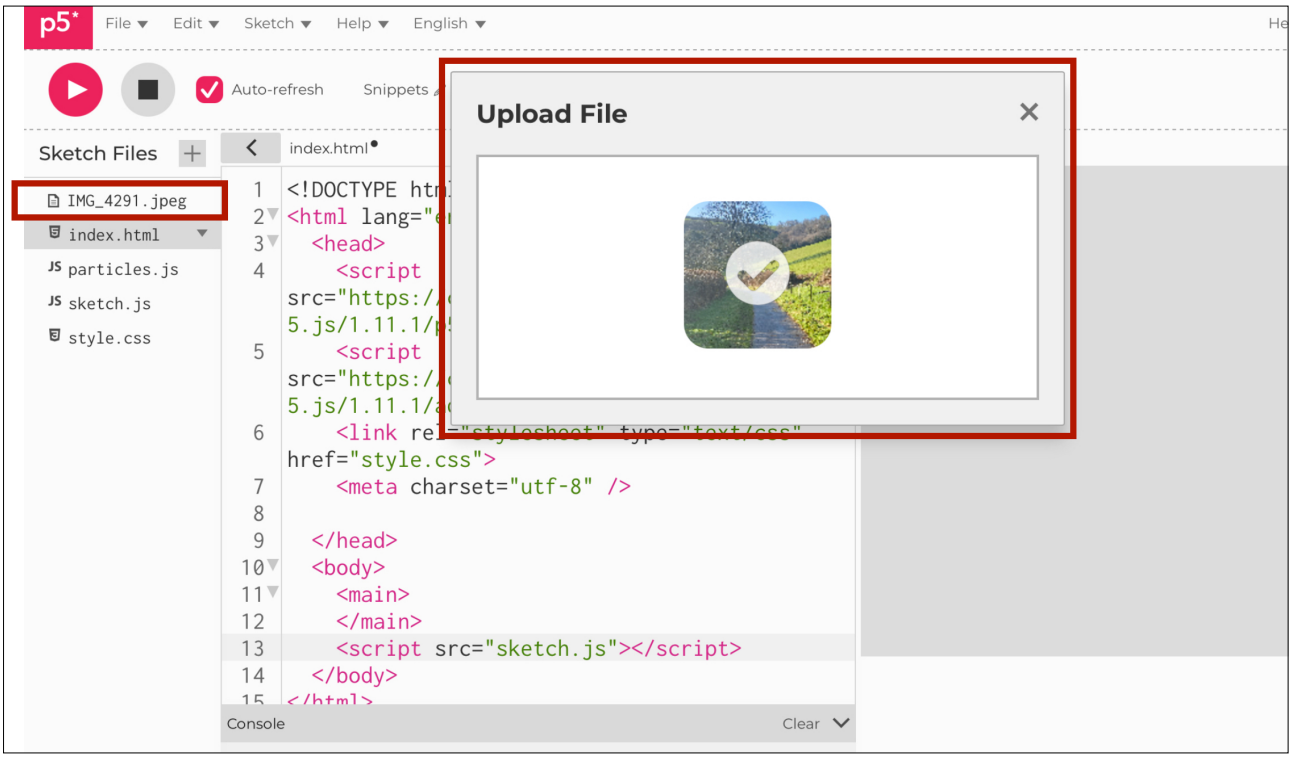
Going back to the `Add files` bit, we can also `Upload files`. These tend to be files that contain images, data, videos, models, fonts, etc. The process is quite simple, but you do need to have the files ready to either browse to or drag and drop from your desktop.

Figure 25: uploading files



1️⃣ In this instance, I uploaded an image of a photo I took while walking one day in Devon. You will notice that the image has a filename on the right in the Sketch files. We can change this to something more memorable. Just note that there is a limit to the size of the file, and the first one I chose was just too big; the limit is 5MB, so make sure you choose a file that will meet those limitations.

Figure 26: uploading image



2️⃣ Let's change the name to `path.jpeg`. It is important to keep the same file extension, but we can change the name.

Figure 27: renaming file

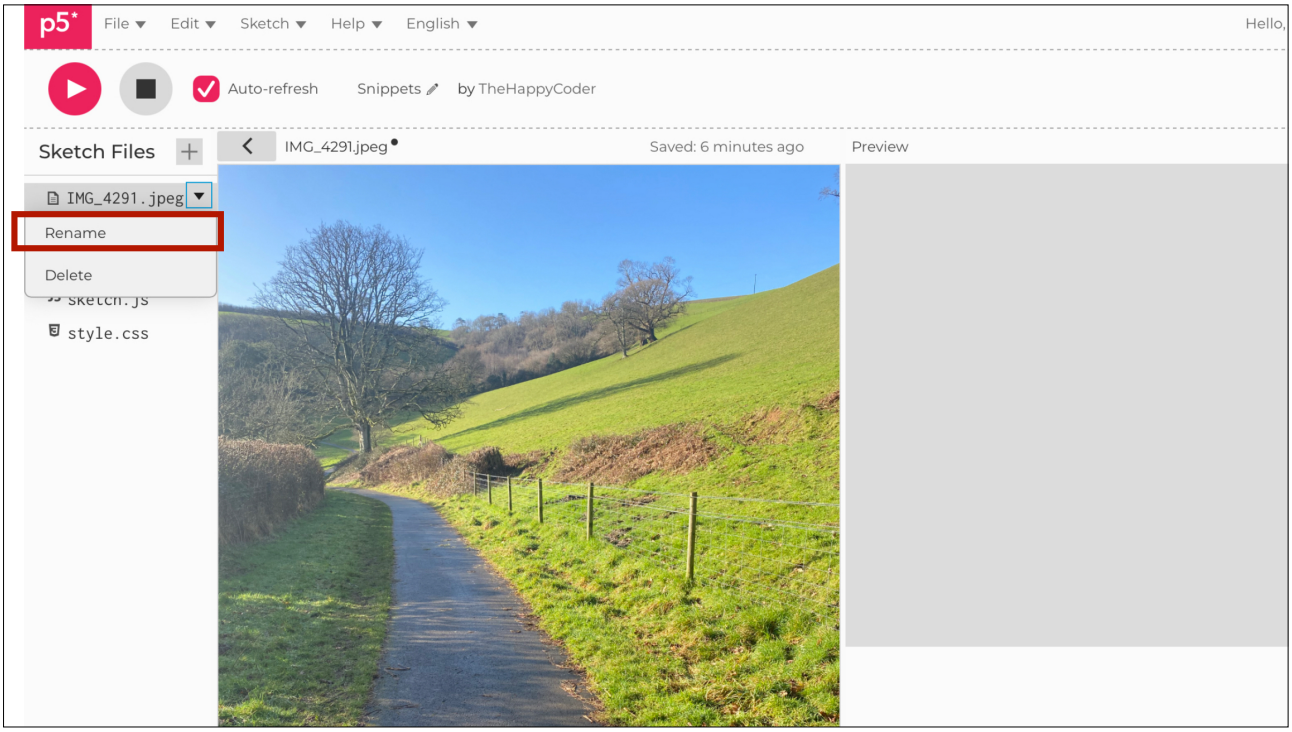
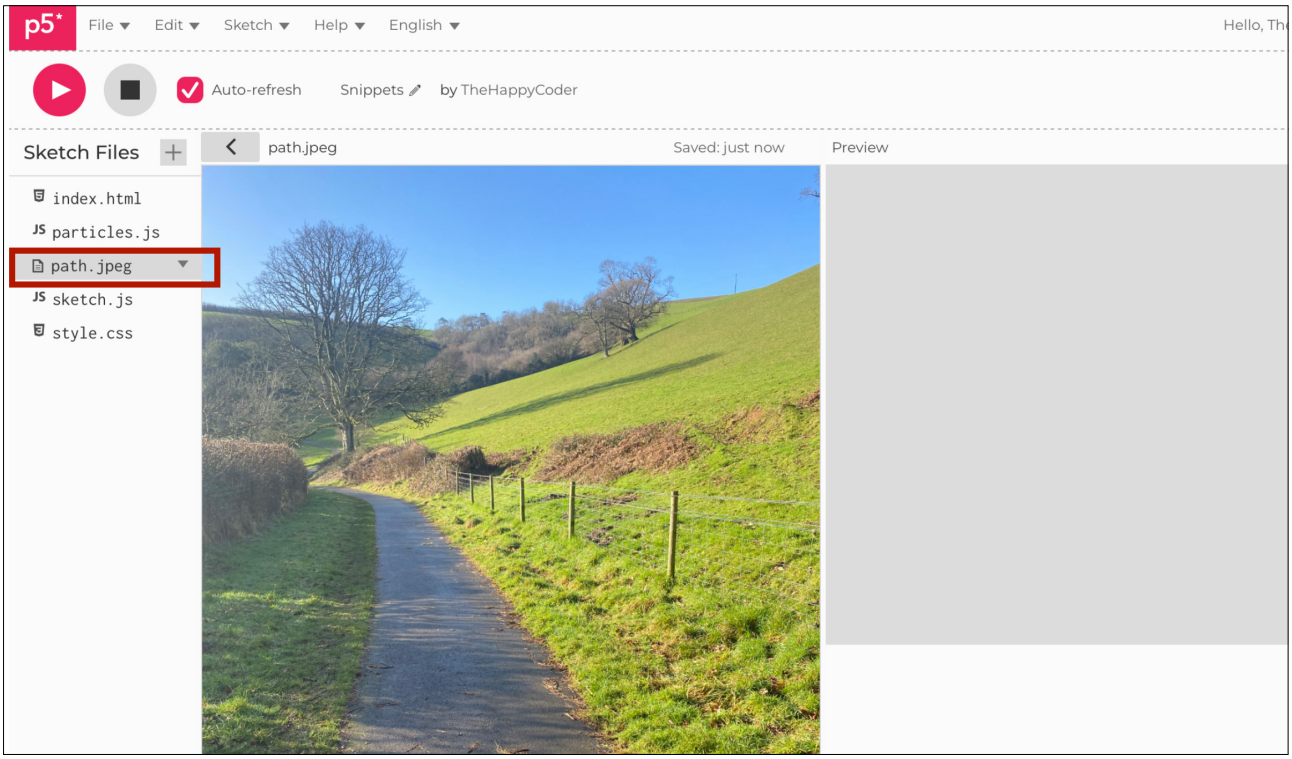


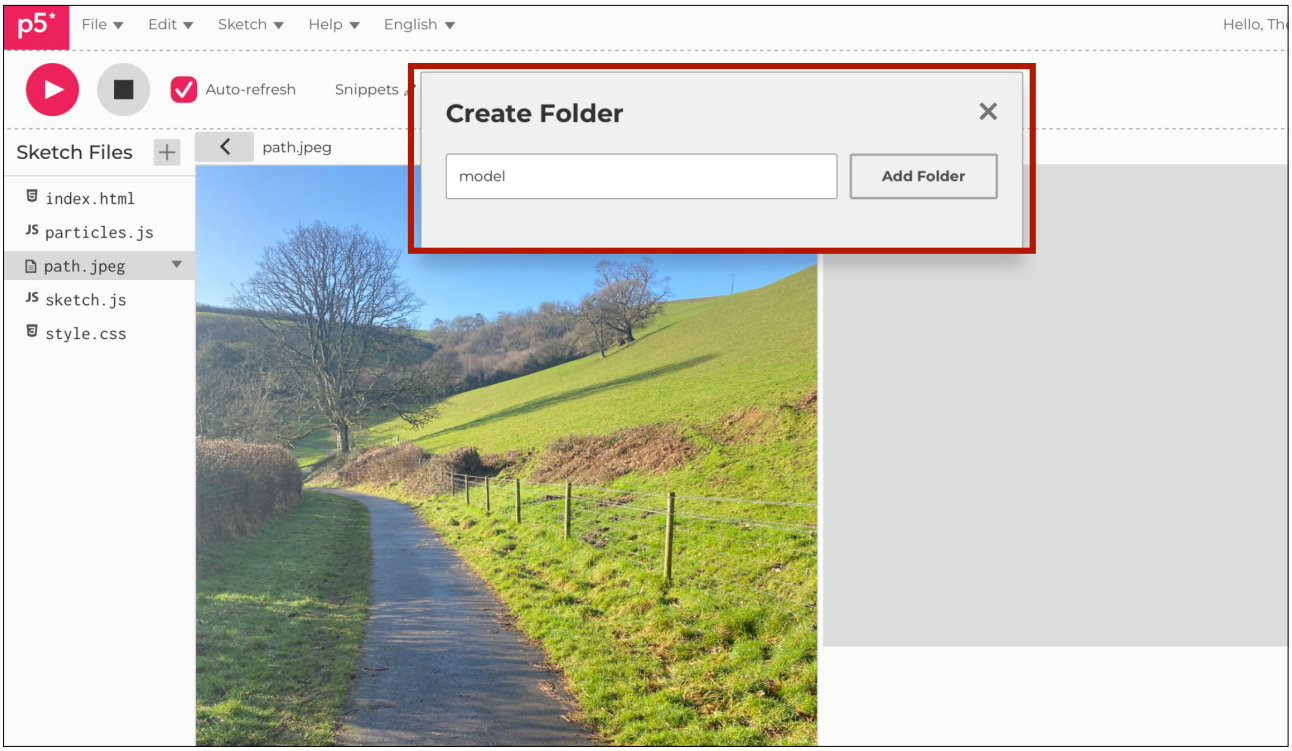
Figure 28: renamed file



Creating folders

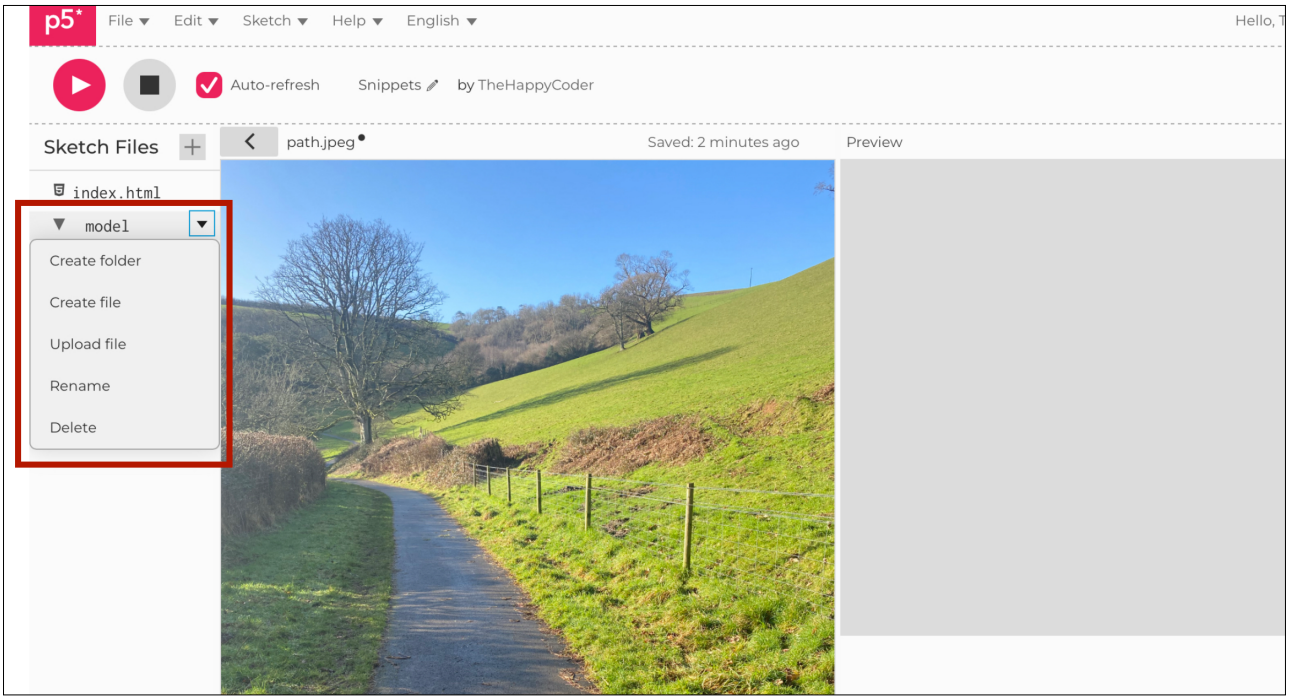
There are instances where you might want to group some images together, or as in the case of saving an AI model, you need to put all the files together in one folder. To do that, we can create a folder and give it a name. Once it is created, we can create files or upload files into that folder.

Figure 29: folders



As you can see, you can even create a folder within a folder, etc.

Figure 30: folder menu



Final words on this section

On the whole, there is a lot to say about the web editor, but most of it is fairly intuitive. The best way is to play and explore, but for now, you can have a read through to see what options are available and the use they may or may not have.

In the next section, we will have a look at ml5.js, our route to machine learning and AI.