Promises

A promise works just like it does in real life, your code basically promises to do something at a later time then when the promise was called and then it can either do it or not. You can then take actions depending on whether or not the promise was kept. This can be useful, for example, if you need to get information from an online database, but you aren’t sure if you’ll be able to connect to it or not. You can have your code fetch that information asynchronously, so that the rest of your website can load in the meantime and the user can interact with it (otherwise, the whole thing would freeze while you wait to get that information.) Then, once the fetching action is complete, whether or not it succeeded will determine how the code proceeds next (depending on how you wrote it).

In this example, if the user’s phone has battery, it will power on. Otherwise, it won’t.

Text

Description automatically generated

Async/Await

You can declare a function as async and then make elements within that function await, and that means that it will not execute the next statement until the await statement is complete. This can be useful if you need to do something with information you get from a server online or something. You can’t do the next steps before you get that information, so by making the “get information” step await, you force the rest of the function to wait to get that information, which is a lot better than guessing how long it will take and setting a timer or something like that.

In this example, a function is written which will print the weather for the user. The code first gets the user’s location, which could take time. Then it gets the weather at that location (which could also take time.) Finally, it prints the weather. Because each step can’t happen until the step before it is completed, location and weather are await.

Graphical user interface, text

Description automatically generated