* Linting and formatting
* Selecting transcript lines in this section will navigate to timestamp in the video
* - [Instructor] Take it from me, the dyslectic coder .
* Writing code without any errors is not easy .
* And writing code that is consistently formatted so all indentations are the same and the code conventions and guides are followed correctly is also not easy .
* To help with all of this, your code editor has a bunch of useful tools built in including automatic syntax highlighting, which is what you see if I open a file here .
* The color coding that indicates the different types of content that are in this file .
* And other features like automagically cleaning up your indentation and so on .
* To help us write cleanly formatted code without too many errors, we can also add to the functionality of the code editor using extensions .
* In this example, we'll add in two new extensions .
* ESLint and Prettier .
* And you can install them both directly from Inside VS Code by going to the extensions option just searching for ESLint and installing it and then searching for Prettier and installing .
* It looks like this, and it looks like this .
* The Prettier code format does exactly what the name suggests, this makes the formatting of your code prettier for your human eyes .
* So that all your indentations are correct .
* So that all your parentheses are lined up properly and things just look clean when you're working with a code .
* It sounds really finicky to use a tool like this but the reality is it makes it easier to work with your code, especially if you're working with other people because all the code will be formatted the same way .
* ESLint is sort of like a spellcheck but for JavaScript .
* It lints, as in goes through your code and finds all the lint and takes it out .
* So it goes and checks your code to make sure that you're following the conventions or the standards that you've applied to the code .
* It's a really useful tool to tell you when you're doing things incorrectly, or when there, standards that need to be followed in a different way .
* And again, it helps you make sure your code is consistent .
* Now for these two tools to work, we need to install the necessary settings and the necessary automated background functions .
* And that's done using Node .
* js .
* Now, remember how I said when you work with modern JavaScript, you end up working with Node .
* js, the JavaScript server runtime .
* That's exactly what we are going to do right here right now .

Text

Description automatically generated

* Oh, and by the way, if you don't want to do this, you don't have to .
* These tools are not required to be able to follow along in this course, but I encourage you to install them because they will make your experience working with JavaScript much more enjoyable .
* And these types of tools are essential to your future work with JavaScript .
* Okay .
* To get all this to work, you need to have Node installed on your computer .
* If you're using Mac OS, you likely have Node already installed .
* If you're on Windows, you likely need to install it .
* Node can be installed directly from nodejs .
* org, where you'll also find all the installation instructions for your operating system .
* Once Node is installed on your computer, we need to install the extensions that allow Prettier and ESLint to work properly .
* Now, these extensions have already been configured inside the exercise files for this project .
* However, you need to install them into your computer .
* That's done by going to the terminal inside VS code .
* To get to the terminal, you can either go to view and select terminal or simply hit control and backtick on your keyboard .
* The terminal is a command line for the computer .
* So here you can do things like install new extensions .
* The only thing you need to do from terminal is once you've navigated to the exercise files folder and you see what I'm seeing on my screen here, type in npm install .
* Now the Node package manager goes onto the internet, finds the dependencies necessary, pulls them into my computer through the internet and installs them in our current project .
* You can see it here at the bottom of my project, I now have a new folder called node modules that contains all of the dependencies that were installed .
* These dependencies are literally dependencies for everything to work .
* In this case for Prettier and ESLint to work .
* So that means when you start a new project, you can install dependencies for just that project .
* If you want to learn more about node and dependencies and npm, we have entire courses in the library focusing on that in particular .
* And I encourage you to go check out those courses after you're finished watching this course .
* But for now, we have all the pieces we need .
* There's one last little piece of this puzzle to get everything to work properly .
* That is to go to file and then to preferences and settings .
* And from here, search for formatsonsave .
* Make sure that this option is checked .
* It's probably not, so you need to check it .
* All that happens when you do this is anytime you save a file, Prettier and ESLint kick in .
* Check over all your content, format the content, if necessary and then save the file .
* Now that we've done this, we can check to see if it works .
* So if you go to the exercise files and go to 01\_05, that's the next movie .
* I can take a preview right now .
* Then you go into script .
* js and you'll see there are no warnings .
* There's no errors in the script right now .
* So let's introduce an error .
* We'll take something away .
* And you can see all these red lines .
* Uh-oh, something's wrong .
* If we hover over any one of them, it'll tell you comma is expected .
* Here will say comma expected again .
* It's because I took away that equal sign .
* So that's ESLint in a nutshell .
* It goes in and tells you, hey, hey, hey, something is wrong .
* Let me fix it .
* The other thing is Prettier .
* So if I reformat my code here, so I'll take away the parentheses and I'll take away the quotation mark and I'll do some indentation here .
* So the code is now messy from a standard's perspective .
* Now watch what happens when I save this file .
* Prettier now kicks in and automatically cleans up my code .
* So it adds the parentheses back .
* It cleans up my indentation and it adds that semicolon .
* And this is the purpose of these tools .
* So if you imagine you work in a team and you have a specific set of coding standards .
* Let's say in your team, you don't want to have semicolons at the end of any line, you make a rule for it and then Prettier will enforce the rules .
* So if someone adds semicolon, Prettier takes it away .
* And, or on the other hand, if you're using a standard that says semicolons should be there and people don't add the semicolon, Prettier comes in and adds the semicolon for you .
* So now with Prettier and ESLint, you have fully automated code cleanup and code linting .