* - [Instructor] JavaScript has some very basic language basics worth knowing about .
* **First, write your JavaScript from top to bottom** because that's how the browser will read it .
* A good rule of thumb here is to define all variables and functions and objects and anything else before you use them, so higher up in the file .
* That way you know the browser has read the object or the function or whatever it is before you call it, otherwise you might run into problems .
* **Second, comments**, like what you see at the top here are used to leave literal comments for yourself and other humans to explain what's going on .
* When you create a comment it is ignored entirely by the browser, so it's purely there for us humans .
* In my code editor right now with the color settings you can see comments are colored green .
* They'll always have a different color from other code and it's always consistent in the way you see here .

Graphical user interface, text, application, email

Description automatically generated

* To leave a single line comments in JavaScript, you use two forward slashes and then just the string of text that is the comment .
* You can do that on its own line and you can also do it inside another line of code .
* So, here we have a declaration for a let and then we have a comment directly afterwards .
* Just remember everything behind the two forward slashes will always be the comment .
* If you need multiline comments, you use CSS comment syntax .
* So, forward slash asterisk to start the multiline and asterisk forward slash to end then you can add as many lines of texts as you want in between these two markers .
* If you are using a code editor like VS Code, there's also a shorthand for a more verbose type of comment .
* Down here I have a function called updates backpack and if I enter forward slash and then two asterisks and then hit return, VS Code will create a verbose comment for me and even capture that this function has a parameter called update .
* It will then ask me what type of data is in updates, I'll say string and then I can fill in this comment .
* And here you see the purpose of a comment .
* Now, when I come back to the code I'll see there's a function here, it's called update backpack, it outputs HTML and it has a single parameter, it's a string and it's called the update .
* Comments are also often used to quickly activate or deactivate single or multiple lines of code for debugging purposes .
* This is done by placing your cursor on the line or lines you want to comment out and then hitting control or command forward slash .
* So, if I hit control forward slash I comment out this line of code, if I do it again, so control or command forward slash, I uncomment this line .
* If I want multiple lines I just highlight the lines and again, control or command forward slash to comment them out, control or command forward slash to take the comment away .
* It's a very quick and easy way of doing debugging .
* **Third, spacing or white space .**
* In JavaScript, spacing is purely for the human user .
* That means the browser does not care about spacing at all .
* You can write all your JavaScript in a single line and the browser will still understand, but humans will not .
* So, it's better to use spacing properly .
* The best practice is to use indentation to visualize hierarchy .
* You can see that down here we have a constant called backpack and then inside we've indented all the pieces that go inside and there's some further indentation to show what the hierarchy is .
* Now, you may have heard a joke or two about tabs versus spaces .
* This indentation is what those jokes refer to .
* You see at the top here, this indentation currently is two spaces .
* Now, that joke used to make sense because you had to add either tabs or spaces manually to do indentation, but today this is all handled automatically by the editor .
* So, if you see down here, it says, "tab size four .
* " So that means if I hit return here and then hit tab, here we have a single tab and it's four spaces wide .
* If I want to change that I'll go into my code editor and say, "I want to indent using spaces," and then say how wide I want the indentation to be, two .
* And then you see the indentation I created is automatically changed .
* So that way, no matter what the indentation is for a document you can always just convert it using the code editor .
* And there's more, if you're using a tool like Prettier this will be sorted out for you, so you don't even have to think about it .
* **Fourth, semicolons,** you'll notice that I add semicolons at the end of all my declarations and you'll also notice that Prettier will enforce this standard .
* Now, some developers will add semicolon to the end of declarations, others will not .
* Again, JavaScript does not care either way .
* This is a coding and syntax preference for us humans .
* In some coding standards, you are required to add coding, add the semicolons, in others, you aren't required to not add them .
* The good news is, ESLint and Prettier will sort this out for you .
* So, if you go look at code examples and you see either using or not using semicolons, know that this is just a preference .
* If anyone ever tells you it's wrong to do one or the other, that's simply not true .
* This is purely editor preference and the tools we use will automatically sort it out for you .
* **Finally, quotation marks .**
* You'll notice, in my code I use **double quotes for all strings** .
* This again, is **developer preference .**
* Some developers like to use double quotes, some like to use single quotes and different coding standards will enforce different styles .
* So, in some **coding standards, single quotes will be enforced** and apply of using ESLint and Prettier, in others **you'll have double quotes and the same thing will happen** .
* So, just be consistent in what you're doing and the tools will figure out the rest for you .
* *The rule of thumb to all of this is, number one, be consistent and number two, use tools like ESLint and Prettier to automate this consistency*