* - [Instructor] In the example project, I used this template literal to create HTML and then added that HTML to the existing document by targeting an existing element and replacing its internal HTML using the inner HTML property .
* All that functionality is down here, and we've now covered all the features that make this possible .
* This is an effective way of injecting new HTML into a document, but it's also rather crude and destructive .
* It requires an existing element to be in the HTML documents, and it wipes up whatever code was contained inside that element .
* And that can be a problem .
* Let me show you what I mean .
* In the exercise files for this movie, I've added a short intro text to the main section of the document .
* This new text sits inside the main element .
* And the problem is when we inject the backpack content using JavaScript, we're replacing all the content inside the main element, so this new text disappears .
* You can see that if you go to the browser the new text is nowhere to be found because we replaced it .
* What I want to do instead, is to add the backpack content inside the main, to appear after all the other content, so after this new div called page header .
* To get this to work, I need to add a new DOM element, and place it exactly where I want it to appear in addition to what is already there in the document .
* So instead of a destructive process, I want to use on process that creates new content .

Graphical user interface, application, Teams

Description automatically generated

* DOM manipulation like this, is an essential part of JavaScript .
* So there's a huge toolkit of methods available for just this purpose .
* So here's what I want to do .
* First, create a new element for the backpack content, it'll be an article .
* Then populate that element with classes, IDs, and content, and finally somehow append that new element to the existing document .
* It all starts with this method here, create element .
* In the JavaScripts, I go down towards the bottom, and I set up a new constant called new article, and I set it equal to documents, creates, elements, and then I just name what type of elements I want .
* So in this case, it's an article .
* This will create a new article on the DOM, but it's not placed anywhere .
* So it kind of exists in theoretical space until we use it .
* Now that we've created this new article, I'm also going to take the article out of the template literal here so we're not doubling the article .
* Then I want to append a class to this new article and I can do that using the methods we covered previously .
* So new article, class list, new and .
* .
* .
* sorry, add .
* .
* .
* and here we want backpack .
* And then I also want to add an ID and we do that using set attributes .
* So new article again, set attributes .
* This is an ID and the value should be every day .
* And finally I want to place the template literal content inside this element .
* I already did that with the main .
* So I'll just change this to new article .
* And that way the inner HTML of the article is set to this template literal .
* Okay, so now we have an element with a class and an ID and content .
* The next thing we need to do is somehow append this new element to the existing content .
* So it is appended to the main to appear after this div with the class page header .

Graphical user interface, text, application, email

Description automatically generated

* That's done using the parent Node append method .
* This method takes the parent Node, in our case the main and then it appends whatever we want to the end of that main as the last child .
* And that's exactly what we want to do here .

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

* So we already have the main defined up here, document query selector, main content .
* I'll grab main and say append, and then append new article .
* All right, save that, go back in the browser and test it and look at that .
* *Now we have the main content and the appended backpack and this is possible because we created a new element and then added that element into our document .*
* ***The append method can be used to add one or several comma-separated elements and they all go at the end of the parent Node .***
* *And it can also be used to append a string of texts* if that's what you want to do .
* *If on the other hand you want to add an element as the first child of the parent, you can use the parent Node* ***prepend*** *method instead, it does the exact same thing .*
* Just places the element at the top instead of at the bottom of the parent element .
* ***If you need to move an element from one location to another inside the DOM, or you need the browser to return the appended object to be further worked with, you can also use the append child method .***
* The only difference is it also returns this element to you .
* There are also several other methods for this type of manipulation and their names pretty much explain what they do .
* Graphical user interface, application

  Description automatically generated
* A picture containing graphical user interface

  Description automatically generated We have replace Child, which replaces a child element of a parent .
* Graphical user interface, text, application, email

  Description automatically generated
* We have insert Before, which allows us to insert an element before the parent elements .
* And we have insert Adjacent .
* Now these two last ones are interesting because they give you the capability of adding new elements before or after other elements rather than just inside them .
* In particular, insert Adjacent Element is extremely powerful because you can specify exactly where you want that new element to appear, inside or outside the target element .