* - [Instructor] An EventListener is exactly what it sounds like***: A method added to a target, usually an element, that listens for a specific event and then calls back a function when that event is detected.***
* In this code example, we have an EventListener appended to a button.

Graphical user interface, text, application, website

Description automatically generated

* *When the button is clicked, we log an event in the console.*
* So the structure of an EventListener is always the same.
* A) We start by grabbing an event target.

Graphical user interface

Description automatically generated with medium confidence

* This can be a window object or the document object or any element within the dom.
* B) Next, we add an EventListener with the addEventListener method.

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

Description automatically generated

* This method takes **two main arguments**.

Timeline

Description automatically generated

* **First, the event we want to listen to in quotation marks**.
* This can be any of the events listed in the MDN events reference, and second, a callback function to run when that event is triggered.

A picture containing timeline

Description automatically generated

* This can be either **an inline anonymous function or a call to an external function.**
* And remember, this is a callback function, so there's no parentheses at the end here.
* So we are actively calling the function into the EventListener and then running it inside the EventListener.

Graphical user interface, text, application

Description automatically generated

* There's also a third optional argument for an options object that gives you more control.
* Now, this is an advanced feature that's rarely used and falls well outside the scope of this course.
* Developers will often set this **third argument to false** to ensure default behavior takes place, but it's not strictly necessary.
* So in most cases, you can just set up a target and a callback and be done with it.
* If we look at a practical example, we can call in a button for example, here, and then we make the button the target.
* Graphical user interface, text, application, website

  Description automatically generated
* We add an EventListener to the button.
* Graphical user interface, text, application

  Description automatically generated

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

Description automatically generated

* We specify the event we want to listen to; in this case a click on the mouse, and then we fire some form of callback functions.

Graphical user interface, text, application

Description automatically generated

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

  Description automatically generated
* So here, I'm capturing the event in the E and then console logging out to the event.
* Now, let's look at this in code to see how it all works.
* In the exercise files for this movie, you'll find our backpack packer site with the new expanded features we added in the challenge at the end of the previous chapter.
* There's only one new thing here.
* I've added a button with the class lid-toggle and the text Open lid.
* And if we go to the front end, you'll see the button sits under each of the backpacks.
* I'm going to click on the button, nothing happens.
* So this is where we want to use an EventListener.
* What I want to do here is create an EventListener that's appended to the button, and when I click the button I change the lid status from closed to open and back to closed again.
* Now the key to doing this is to make sure the EventListener is appended to each of the articles independently.
* And this code is set up so that we loop through each of the backpack objects in an array.
* And then we output a separate article for each of them.
* And that's done using this function here.
* Now the article itself sits inside the backpack article variable.
* So that's what we'll target throughout.
* Inside our loop, before we return the backpack article to the code to be output in the browser, I'll first look for the button.
* So I'll set up a new const called button, set it equal to, and then I'll grab backpackArticle.
* That's the entire article.
* And then I'll use querySelector inside that article.
* And I'm looking for the class lid-toggle.
* So that's the button up here.
* Now I can append an EventListener to this button.
* I want to listen for the click event.
* And when the click event occurs, I fire an anonymous arrow function.
* Here, I want to capture the event itself so we can see what it is.
* Set up an arrow function.
* And inside the arrow function we'll just console log out the event.

Graphical user interface, application

Description automatically generated

* Save that, go in the browser and open the console and click on the EventListener button and now, we fire an event and you can see every time I click it, we get a new event.

Graphical user interface

Description automatically generated with medium confidence

* Now, if we open the event object, you see there's a ton of information about the event here.
* What I'm looking for in particular right now is the path.
* The path gives me the dom path for the event that was triggered.
* So here we have the button.
* Then we have the article, main, body, HTML document.
* What I'm looking for is this one here: articlepack1.
* That's the ID for the current article.
* All I want to do is make sure I have two separate events.
* So if I scroll down and click the other event and open this again, scroll down, you'll see here we have articlepack2.
* So that means we have two separate EventListeners firing on two separate buttons.
* All right, now I want the button to do something.
* I want to change the text up here from open to closed and back to open.
* And we can use a turnery operator for that.
* But first, we need this element.
* So I'll set up a new const.
* Status.
* Set it equal to backpackArticle again, querySelector again.
* And then this time, we're looking for the L-I that has the class backpack\_lid, and we're looking for the span inside that backpack\_lid.
* So span.
* Then inside the EventListener, we'll say status.
* innerText === "open.
* " So that's what we're testing for.
* And now we say, if that's the case, then we set status.
* innerText = "closed.
* " Otherwise, or else, status.
* innerText = "open.
* " All right, save that.
* Back in the browser, click on the button and watch up here on lid status.
* Click the button, it changes.
* Click the button, changes again.
* All right.
* So now our EventListener works and you can see an EventListener appended to a button can influence anything else in the dom.
* That's kind of the whole point.
* We now have complete control over everything that happens in the dom through an event that we are tracking ourselves.