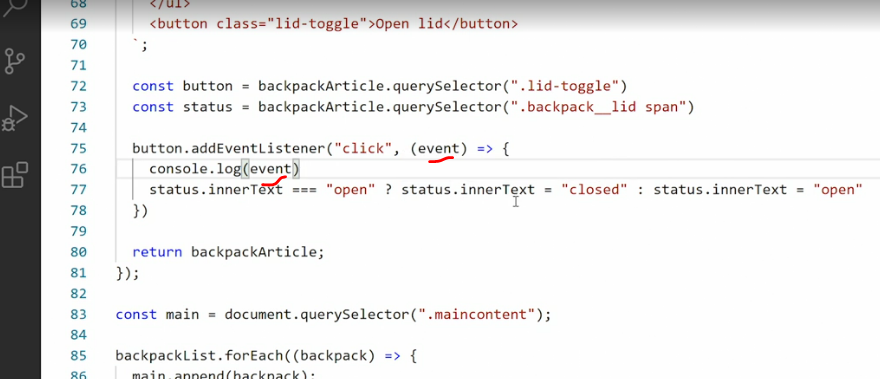
* - [Instructor] Working with the event listeners **you'll often want to either access the event object inside the callback function** or **pass some argument through to the callback function.**
* It's not immediately obvious how to do either when we are working with callback functions.
* So let's take a closer look.
* In the earliest example of an event listener, you may remember*, I called in the event object as a parameter in the arrow function, and this is the answer to the first part of the question.*
* You see, *whenever you set up a new event listener and you have a callback function,* ***the event object is automatically passed to the callback function as an argument***, and **we can choose to capture that object and use it in the callback function by setting up a parameter name for it.**
* So in this case, I've set up the name event and then we use it down here.

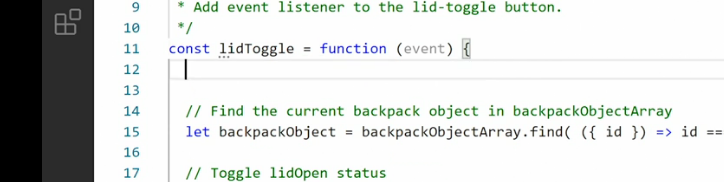


* The *passing of the events object happens automatically.*
* And this is actually the answer to how to do it externally as well, because this entire thing here is just an *anonymous callback function.*

Graphical user interface, text, application, Word

Description automatically generated

* What we have here instead is we're *calling in, the callback function*, but what we're really doing is saying *take the entire lid toggle function and just put it where the lid toggle keyword is.*



* So if we want to gain access to the events object we just need to call it event and then console, log event, save that, go check in the browser and everything is working as it did before.
* And that's really all there is to it.
* If you just want to get ahold of the event object, simply name it as a parameter and use it, and you have it available to you.
* But **what if we want to pass a proper argument into the callback function**? Let's say I set up a new argument here and set it equal to some texts.
* *Looking at the code we have, you can clearly see I can't add a parentheses here and then pass the argument directly because then I break the code.*
* So I have to do something else and what I'll do is quite literally cheat.
* So right now we're using lid toggle of the callback function.
* I will change that, so **instead we set up an anonymous function at the callback function and that anonymous function will wrap around lid toggle.**
* Now we can call lid toggle as a proper function and pass whatever we want to lid toggles.

Graphical user interface, text, application, email

Description automatically generated

* So now I can pass newArg directly to lid toggle and then up here in the function, I need to change this to newArg and console log out newArg.

Graphical user interface, text, application

Description automatically generated

* Did I spell that correctly? Yes, yes, yes.
* Okay, save that, go back in the browser and click the button and we get the argument but we also get an error cannot read property parent element of undefined.

Graphical user interface, text, application

Description automatically generated

* So what is happening here? Well, first off we were able to pass the arguments through to the callback function, no problem *but we also broke our callback function in the process*.
* Why, was because in the *callback function you'll remember we are using the, this keyword that this now refers to the internals of our arrow function.*

Graphical user interface, text, application

Description automatically generated

* So that's not going to work.
* That means ***we need to pass the button element on as an additional argument and then use that instead.***

Graphical user interface, text, application, email

Description automatically generated

* So we have the button element here, we can pass it on.
* So now we have button and you are guests arguments that means we need to have two parameters up here, button and newArg.

Graphical user interface, text, application

Description automatically generated

* Then I'll just change all of these references to this two button instead.

Text

Description automatically generated

* Save that back in the browser, test it and now you can see everything is working again.
* Cool, so now thanks to the cheat we're able to pass any arguments we want into the callback function, but **what about our event object?** Well, the answer is straightforward.
* We capture the event object here, just like we did before.
* This just makes it available to us and then we pass the event object as well.

Graphical user interface, application

Description automatically generated

* So event, button, newArg, scroll back up to the function, say events, button, newArg.
* And then we can console log out the event.
* Click the button and now we have both the event and the arguments passed through.
* This year is a very common pattern for using event listeners.
* We are working around the problem of not being able to pass arguments into a callback function by using the callback function as a function wrapper to call other functions.
* That way we have full control, we can do whatever we normally would do in a function call and we still have access to the event, and we're still using the event listener as intended.