CS 240: Lambda Syntax Transcript

[00:00:00] **INSTRUCTOR:** Now that you know how lambdas work, you should understand some of the details of the syntax. So, here's what the syntax looks like.

[00:00:09] We have a parameter list, then we have an arrow symbol, and then we have a body.

Start slide description. Following text is shown:

(Parameter List) -> Body

Parameter List

- Comma-separated list of formal parameters
- Data types are optional and can be inferred unless ambiguous in callee
- Parentheses are optional for a single parameter
- Empty parentheses are required for an empty parameter list

->

- Arrow token required between parameter list and body

Body

- A single expression or a statement block
- Return is inferred for a single expression
- Statement blocks are enclosed in curly braces
- Return must be explicitly specified (unless void) for statement blocks

End slide description.

[00:00:15] So, let's start with the parameter list.

[00:00:17] So, there are a few details that you should understand about that.

[00:00:20] So, first of all, it's a common separated list of formal parameters.

[00:00:23] Formal parameters are just the parameters that we can declare and use in our code.

[00:00:28] So, we refer to them throughout our code. So, it's a common separated list of those. Data types are optional.

[00:00:35] They're usually not specified because they can usually be inferred, and the whole point of using a lambda using lambda syntax is that it's brief and simple.

[00:00:43] So, we don't want to complicate it with extra syntax that we don't need.

[00:00:47] So, we typically don't specify data types in this parameter list, although we can if it's ambiguous for some reason.

[00:00:55] Sometimes we are calling a lambda expression that we're creating a lambda expression that only has one parameter.

[00:01:02] And in that case, the parentheses are optional.

[00:01:04] So, if we have just one parameter, we could leave off the parentheses and just specify the variable before the arrow.

[00:01:12] And then another thing you know about that is if I'm not passing any parameters to my lambda function, then I have to specify empty parentheses.

[00:01:22] If I just leave a blank, it's a syntax error.

[00:01:24] So, it's empty parentheses that indicate that this lambda function doesn't take any parameters. So, that's the parameter list.

[00:01:32] And then we have the arrow token that indicates that this is a lambda expression.

[00:01:37] Now one thing that's a little bit unfortunate is that this arrow token is different than what's used in a lot of other languages.

[00:01:44] So, for example, in JavaScript and TypeScript, we have an arrow, but the arrow is an equal sign followed by this greater length than simple.

[00:01:53] So that might confuse you if you're used to JavaScript or TypeScript.

[00:01:56] We'll just have to get used to it that the equal sign here doesn't work in Java.

[00:02:02] Then, we have the body, and there are a few things to know about the body.

[00:02:05] So, if we just have a single expression and that needs to be executed as our lambda function, then it can be either way we can have a single expression, so one expression, what would normally be one line of code followed by a semicolon.

[00:02:20] We can have that. Or we can have a block of code. So, if we have a block of code, we just have curly braces there with all the code in it.

[00:02:28] If it's a single expression, we don't put the return because that's inferred.

[00:02:33] If it's a statement block, then a return is not inferred, and so we have to put a return on the last statement if we need a return.

[00:02:41] If the lambda function is returning void, then of course we wouldn't have a return.

[00:02:46] So, those are the details of lambda syntax.

[00:02:50] You'll get used to those as you work with lambdas, and it'll become second nature.