All right.

So now that we can read tasks, we can add tasks.

Now it's time to figure out how we can check off a task.

Now the actual checking off happens in the task title.

And we need that to be reflected inside our task provider to check off the correct task.

And then we need it to be reflected on the task list.

So the task list is going to be the one listing for that change.

Now there's a little bit more to it than just that because in this case we want to keep the task tile　relatively simple.

So what we want to keep it as a stateless widget and we don't want it to know about all these things　like the data provider or the provider package.

What if we could just have it as a sort of dumb widget which just gets information from the task list　and renders a task tile as and when needed?

So how can we get that checking off behavior while keeping the task tile　a very simple widget without it being involved with the provider?

So firstly notice how currently our task tile is a stateless widget.

It gets passed a isChecked boolean, a task tile string, a callback to notify the parent when the checkbox　gets changed. And then inside the parent which is the tasks\_list.dart　well here we used to have a set state which exists inside the checkbox callback to update our tasks.

Now in this case we don't really have set state anymore because we're no longer inside a stateful　widget.

So how can we keep this task list being the place where we update the state because we already have　access to our consumer widget　so we already can modify our task data object and how can we keep it all limited here keeping this widget　being the complex clever sort of conductor and keeping our task tile simple and just implementing the　things that it gets told to do?

So similar to a large company, you might have a middle manager which is in this case the task list then　you might have an employee which just does what the middle manager tells it to do without it needing　to decide and be aware of the bigger picture.

So keep its job quite simple.

So how could we achieve that using what we know about provider? Well the first thing we're going to do　is we're going to delete this set state inside our checkbox callback. And instead we're going to do that　inside our provided task data.

So inside here, we're going to create a new method. We'll call that updateTask.

And in this method we're going to pass over the task that needs the updating.

And when we get hold of it we're going to call task.toggle Done.

And that if you remember comes from our task.dart　so when we defined how a particular task should behave we created a property called isDone which can　be switched to the opposite of itself　using that toggleDone method. And that's what we're using right here.

But once we're done toggling the task property isDone, then we're going to call that really important　method

notifyListeners to update the widgets that are listening to that particular task　state so that they rebuild and reflect the current doneness or the checkedoffness of that task.

Now all we have to do is to use it. As I said we're going to keep our task tiles simple.

So we're going to keep it exactly the same.

It's still going to have a callback,　it's going to report back to its superior which in this case is task list and it's going to have this　callback which it triggers whenever the user changes the checkbox.　So when it checks it off or unchecks it. And inside this callback is where we're going to tap in to our　task data and call that method update task. And the task that we're going to pass over is going to be the TaskData.tasks and it's going to be the one at the current index.

So the task tile gets built from our list view builder and for each item in the task tile, it gets assigned an index.

So zero will be the one at the very top and then 1 and 2 etc..

So in order to get the right task to update, we also have to use that index.

But notice how we're using this taskData.tasks[index]... now in three places.

So we can actually refactor this quite easily by going just above the return statement creating a task item and we're going to set it to equal taskData.tasks at the current index.

And now that we have a hold of that task we can use it in all of these places.

So it'll be task.name, task.isDone and also task is going to go inside here as the one that needs to be updated.

So this also simplifies our code and keeps it short and sweet.

So now if we hit save and we rerun our app, then we can go into our Todoey app, let's just add a new task and we can check it off or check off any of the other ones and we can uncheck them as we need to.

So now all that's left is, what if we wanted to delete a task because at the moment even when we check them off it just says it's done? And we don't want to just check it off and delete it because we don't want to deprive our users of the joy in checking off a list making sure that everything is crossed off.

In fact sometimes I think I just do things to check things off my list and it looks really nice when they're all ticked off.

But what if they wanted to clean up this to do list?

Well, what if we could get them to click and hold on one of the items and then it just deletes it out of the list?

So that's what we're going to tackle in the next lesson.

So for all of that and more, I'll see you there.