





In this lesson, I want to introduce events and events, make up the third possible component of a class

and instance objects of a class to you, the two components, they are attributes and methods.

And you've already seen those in a lot of detail.

So let's ask that question, well, what are events?

Well, events are really a concept that allows for an object or class to publish an event while another

object or class can respond to the events.

And in our body, we use the key word event to help implement this concept.

Now, as we've seen throughout the lessons in this course, a method of an object is executed after

a calling program calls it.

And because of this, the calling program and the cold object are considered closely coupled.

They have a close relationship.

Now events are a little bit different.

They are used to decouple the calling program from the cold method.

They have a much looser relationship.

Now, before we go on and explain this event concept in more detail, I need to define a few terms.

First of all, we have events, the world events itself.

We had events as components in a cause, then any method with the class can trigger the event.

That's pretty simple.

We then have 100 methods.

We can declare the methods in the same class or another class to handle a response to the triggered

event.

Then we have the term registering event handlers.

When we have handled methods to objects at runtime, they can be registered by the system as handlers

for those objects of the classes which can trigger the event.

Now a class can implement handler methods for the events of different classes.

And conversely, a class can declare different events in which the object of many different classes

can be interested.

So that last line was very high level.

We're now going to start digging a little bit deeper and look at the syntax and how we use events in

up.

And first of all, we'll discuss triggers to create a relationship between objects and events, you

must implement code for both the classes that trigger events as well as the classes that handle the

events.

And we start by declaring each event, which is to be triggered in the methods of a class in its declaration

part.

And just like methods and attributes, they have to belong to one of three visibility sections of a

class, which in turn will then determine what type of color can handle the event.

For example, if an event is defined in the protected section of a class, the event can only be handled

by objects of its own class or objects of its subclasses.

Let's have a look at how we declare an event.

First of all.

And first of all, you'll notice I have two ways of declaring an event, the first one is an instant

event, just like a normal instant class, and the second is for a static event, just like when we

declare static methods and static attributes.

And so.

Instance, events are objet, triggered events and static events are triggered events.

So if we go through the syntax step by step, we can see we start with the events Keywood, then we

declare an event name.

Then we have the option of using exporting parameters.

An important thing to note here is with events we can only use, exporting parameters we can't use importing

or anything else like that.

It has to be exporting.

And when we use exporting parameters, we always have to pass by value, not by reference.

So once we specify an exporting parameter here, we're using it and declaring it of Type II.

We also have some additions of optional, which no guesses what that means.

Yep.

Is an optional parameter and we can declare default values.

If we have a look at static events, it's almost identical, the only differences we include the word

class and dash before events.

So which class events?

And then we specify exactly the same.

When we want to trigger an event, we make use of the race event statement in the methods of the same

class where we declare the event.

And the syntax is pretty simple.

It's just race event and we give it a name and we have an exporting parameter and then we specify our

parameter and assign a value.

As usual here we're using is the exporting parameter of the event and V is the value we are placing

inside the parameter.

We can use this syntax in all instanced methods to trigger all instanced events and static events of

a class.

But if we use this syntax just in a static method, only static events can be triggered.

Now, the race event statement is the piece of code that forces the runtime environment to Paul's execution

of the method the statement is in, and they execute all the registered handler methods.

Now, once all these registered handler methods have finished executing, processing then continues

in the method that raised the event in the first place.

You can see this diagram on the right here we have the raise events, which forces the runtime environment

to pause this method and then in a sequential fashion, it executes the handover methods one at a time

until they are complete.

And then processing will return back to the method that raised DEVANT.

Now, for us to really get to grips with events and how we use them in our coding, it will be a lot

easier to explain them in a program itself.

So what I've got here is a very high outline of a program that I want you to create alongside me that

will explain how events work.

So first of all, what we're going to do is create a program that focuses on a restaurant and all good

restaurants have a chef to cook the food.

So what we're going to do is create a chef class.

Now, when we create this class, we're going to create an event called Call for Waiter.

So when the food's ready, he can call the waiter and the waiter can deliver it to our customer.

Then we'll declare a method called call service, and this call service method will contain the code

that will actually raise the call for waita events.

We're then going to declare a customer class and it's the same sort of thing, we're going to declare

another event called Call for Waiter.

And please note, this is actually a separate event, even though I've called them the same.

I want to show you that because it's in a different class than it is a separate event because it's contained

within the class.

More than likely you will call it a different name, but for example, we're going to call it the same.

Then we're going to create a constructor method in our customer class that will import a table number.

And the table number is just going to be a simple parameter declared as an integer and then hangs with

the chef class.

We're going to declare a normal method.

And we will call it call for assistance, and inside this method, we will raise the call for Waita

event that was declared.

Now, just before we get onto the op ed, just note that we are not defining any event handlers at this

point, I want to take it step by step.

So, first of all, we'll create a normal program and we'll just implement the code to show how we create

events and how we implement the syntax to raise events.

So let's go over to the op ed and get coding.

So first of all, I've defined a brand new program, and if you don't want to watch me Tiepin, by all

means, feel free to skip forward in the video.

But for those that do want to watch, follow along, see if you can create the program yourself.

And after it's complete, have a play around and see if you can read things yourself as well.

See if you can create your own methods and own events to extend the program.

So I'm going to start out by declaring a framework for the classes, so I'll start out with the chef

class.

And I'll declare the customer class.

So that's the definition, then I'll do the same, but declare the implementation.

So we got that in place and now let's get some code.

So first of all, the public section.

And then in here, I'm going to declare a method.

All coal service.

And now I'm going to declare an event so you can see it's pretty easy stuff.

And that's it, that's the definition for the chef class, let's move down to the customer now, declare

the public section and the methods.

And first of all, we've got to construct a method.

And that's going to be importing.

I value.

And we're going to declare an integer to represent the table number, so let's use table number.

Troops need to declare it as a type I.

And I'll declare another method called Call for Assistance.

And then I'm going to declare the event for the customer.

And this one is going to be called for Waita as well.

Now, this one is a little bit different for this method, we're going to be putting or exporting natural

table number, so define the exporting addition.

And then it's going to be buying value again.

And I'll create E for exporting.

Table number, and this is of Type II.

Testing one last thing here, I'm going to declare table number as an attribute of this class, but

I'll declare it in the protected section so that any external programs cannot access it directly.

I could put it in the public section, but let's stick with these good practices.

That's it.

That's the customer class classifying.

Now down to the implementation.

So if you remember, for chef, we have a method called coal service, look at spelt it wrong, didn't

I?

There we go.

So back down to the implementation to create the code, the code, say, for this method.

Let's do a safe, hopefully I'll pick up my my syntax then.

Now, for this method, first thing I'm going to do is write out a line to the screen and I'll do this

in a few different places just so that we can keep track of how our methods and events are called,

exactly what is happening in our program.

Just like we've done in many others in this court.

So let's say the chef calling waiter events

and was saying that because the very next line that we're going to do is actually raise the event's.

So we say raise events and then we have to give it the event name, which is called for Waita.

This is the one.

And once again, we're right to the screen.

And say, chef coaling waiter event is complete.

And then just to make sure we have some nice separators, I'm going to put on an underlining place and

then that's it.

That's the method complete and we'll go down to the customer method.

What I'm going to do is just a bit of copy and paste in to save me typing.

And if you remember, with a customer, we have a constructor, first of all, that I need to create.

So let's do that.

And that's going to be a simple just assigning the table number

equal to the important parameter.

And then for the call for assistance method, once you get there, where the cursor is going to do a

similar thing as the chefs call for service method, so it will reach out to the screen, let's put

the correct text in here.

Customer calling.

Weightier events than we will raise the event, but has this event actually exports a parameter, as

we've defined a bit higher up here we go, then we need to add that in here.

So we'll will say raise event call for waita exporting.

And we're going to export our table, no.

Which is set.

To our attribute table number and once the event has been raised, will write to the screen again.

Great.

So that's the customer implementation done.

Now, while we're left to do is write the code for a calling program to be able to create these classes.

So just tidy things up with a pretty princess.

Oh, I can split these onto multiple lines, never mind.

The that's part of the program needs to be calling methods, so now we have our objects created, let's

call the methods for those objects.

So I say call method.

Great, so say that I'll do a quick check for errors and then let's go through the code.

Now, I do have an error.

Let's have a look.

Method call for service does not exist.

Oh, it looks as arbitrary.

Renamed it wrong.

They should be call service, not call for service.

Scroll back up.

Yes, a decolletage call service.

We go the check, right, and everything looks good, so I will activate it and I'll run it and you'll

see and I'm going to do that much it just out put these entries to the screen to show that when the

methods are called for each object, those methods will write to the screen.

And because we have Jeff calling Waita event and calling wait for event complete, that indicates the

program run through our method and actually did raise the event.

You want to do it in debug?

Let's have a look at that.

So that's a breakpoint there and then we'll step through it.

So the first call comes up to our chef call service method right to the screen, raises the event.

Not much to say, I'm afraid.

And the last one is right in that complete line to the screen, come back to the customer object, call

for assistance, raise event again and so on.

So that's pretty straightforward, how we define the event and raise the event and the coming videos,

we're going to show how we then take a stage further to implement the complete cycle for events so we

can actually see working.