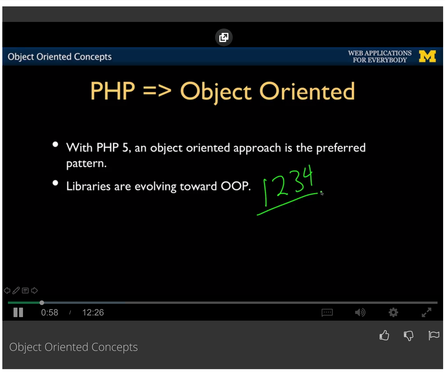
Course Materials

Lecture Content

Bonus Materials



Have a question? Discuss this lecture in the week forums.

Interactive Transcript

English

0:08
So, welcome to our lecture on Object-Oriented Programming. I both love and hate teaching Object-Oriented Programming. I love teaching Object-Oriented Programming because it's a really awesome technique. I hate it because I don't want to confuse you. And so, whenever I teach you Object-Oriented Programming, I don't really give you an assignment that says go write an object which is traditionally the first thing that you do. I really just want you to understand some terminology so in this lecture you can kind of sit back and relax and just absorb because I'm not trying to give you a skill that you can apply but instead some words that I can use later. Okay? So, this is as much a terminology. So, PHP as a programming language, as I mentioned before, has been around for a very very long time. There's PHP 1, PHP 2, PHP 3 and PHP 4. PHP 4 was around for a really long time. it was kind of the awesome one. And then there was a PHP 5 and then remember 6 PHP never happened and then 7 which is the current PHP. 5 is the place that they brought object-oriented or we call it OO, it came in. And so, PHP is sort of of two minds when it comes to object orientation. You see some things that are from the old days, the early old days, and C and Perl were the influences here, right? So that you see a lot of ways that we used to accomplish things in C like prefix all the functions with str or prefix all the functions with date, str_, date_ or whatever. This is kind of a non object-oriented way of coping with complexity as you get more and more complex. And then, as in PHP 5, and the nice thing is we're sort of one beyond it now. So, in PHP 5, things were really in transition but really in 7, object-oriented is very natural and advanced PHP which we'll touch on at the very very end of this class is really becoming quite beautiful and the whole PHP community is very much depending on object orientation so over time, it's more important to know it. But for now, we're just learning the basics and we're not going to write a lot of code that makes new objects. For now, we're just going to be using them. So, just understand it's a language in transition. So everything is a pattern; I talked about Model View Controller, it's a programming pattern, it's a way for us to talk about each other's code, share code with each other and go, Oh, I get what you're doing there, right? If I just give you just a bunch of random code, you're like, Oh man, I've got to struggle to figure all this out. But if I say, Oh yeah, that's an object, you like, Oh, I got it, I know what an object is. If you think about it, any program that you build has some code, loops and if statements and then some data, maybe some arrays, maybe some key value arrays or linear arrays. So you think of the data as one half of the solution to a nerdy problem and then the logic as well. In Object-Oriented Programming, what we're doing is we're sort of taking that notion that there's data and code and we're making smaller granularity of it. What we're doing, instead of saying we have one program that has a data and code, what we're going to say is there's going to be lots of objects and each one of these objects has data and code in it and we draw kind of a bright boundary. We've drawn bright boundaries around, like, functions and stuff but now, we're going to take an object that's a set of functions and a set of variables data inside of it and going to draw a boundary around that. And that's also called encapsulation, it's called isolation. It means that you don't have to look at all the code. That's the beauty of object-oriented, because it's containing a whole bunch of functionality that's very complex in a way that's beautiful. And when you start writing them, when you start writing libraries for others, you'll be, like, Oh, I'm going to spend a lot of time building this really cool library and I'm going to hand it to you in the form of five objects. Don't look too closely inside of it because it was hard for me to write. It's just a way to capture and encapsulate information and make it as usable as possible and only share the necessary details with others. Like I said, this is sort of terminology. We're going to learn the word Class, we're going to learn Method, Object and Instance and Attribute as well. I love this little picture. This creative commons picture of the idea of a cookie cutter and cookies, right? A Class is a template. It's a way to make things. It's not a thing in itself, it's not a cookie. A template is a way to make cookies of the same shape. So then, with that template, you can stamp out as many cookies as you want. And then each cookie can have different attributes; colors, what kind of frosting you put on them. And so these are Objects and this is a Class. A Class is a cookie cutter, an Object is the cookie itself. The template is not, itself, a cookie meaning you can't eat the template. You could if you were like a Transformer or something, maybe you could eat it but don't eat the template. Okay? And then Methods are things inside and Attributes that I don't have on the slide, they're also things inside. There's data inside each one and code inside each one. The whole snowman, cookie model falls down a little bit there. So a Class is a blueprint. It's a generic. It is not, itself, a thing like dog. You might have named your dog Dog and then you actually have an instance of a dog named Dog but, in general, we don't name our dogs Dog but we know what a dog is but there's not a particular dog. An Instance is when you have many dogs and each one has names, Spot and they have different color, fur color, and they, well, they all bark but they might have different fur color. So that's what an Instance is. The word Instance and Object are the sub-religions of the object orientation pattern, use different terminology. You should think of them as the same. An Instance is a thing, an Object is a thing, Class is the template. Different programming languages tend to use these different words but they use them for the same thing. A method is a bit of code that lives inside an object. So if we start saying, here is an object, in this object there is data and then there is code. Method is one of the things that's code and so this object might have five methods, they are different functions. These look just like functions. They're just slightly special. Okay. It's some code that's part of an Object is a Method. So if we take a look at some PHP documentation, and one of the things I want you to know how to do is to read PHP documentation. If we look at sort of a classic way of handling a series of date functions because in the old days, with PHP less than 5, we had to create all the libraries were just global variables and so, just to keep ourselves from going crazy and not having a function called add, and takes like two things, like, what does that do? We just prefix them. You'll see this obsessive prefixing. Because these are all global functions, date_ was a convention. We didn't have to work that way but smart programmers, like, I think I'll just name everything date_ so that I can have a thing called date_add, add, string_add, array_add, etc. That way, I know that those things I have to do with dates and it's my way of like reading documentation. I can say, show me all the date functions. Oh, thank heaven, they've got a date_ prefix. This is kind of like a crude but effective way for organizing the naming space of libraries. Object-Oriented really cleans that up. What Object-Oriented does is it creates a box that not only has code and attributes but it also has what we call a namespace. You can create an add function but that add function lives within this class, the DateTime class. So it's not just add, it's DateTime.add or DateTime whatever and so let's add within that. And that way, you can name the add exactly what you want it to be. You say, Oh, this is going to add two strings together or whatever, right? So it's going to do a thing but we're knowing that this has to do with dates, right? That's the idea that when you go look and you'll see look here when I told you, there are all these things tell you when this is, well, it means that this has to be greater than PHP 5.2 and then, all of a sudden, this DateTime class is there. And then you can make DateTime objects and use those DateTime classes. So you'll see pretty quickly, you'll say, Oh, old and new, right? The OOP Object-Oriented Programming stuff is the new way of doing business. So we look at how this works in terms of code. You are just calling global functions with, you know, big long names, date_default_timezone_set. That's something you have to do, you just have a time function as a global function, date. That's what the current time is and then we're going to add, time gives you back seconds. Since 1, 1, 1970 and then we're adding seconds so that's seven days a week, 24 hours a day, 60 minutes, 60 seconds and so that's the number of seconds in a week, so that's next week. And we're going to do date and we're going to give it a format as the first parameter and if we want it, that's the default will be Now and then Next Week is, if we want to take this variable, that's seconds, and print it out. So that will print these two things out which is Now and Next week. I wrote these slides a long time ago. Very long time ago. That tells you how long Object-Orientation has been inside PHP. Now, if we do this with an Object-Oriented pattern, we have this thing called new. So new is very important. New is the act of taking the template and stamping out a new cookie. So, bam, make a new cookie. You're telling the DateTime class to make a new object using the DateTime template and then give me that object back in the variable $now. Go make a new DateTime, build her all up, give it to me back and I'm going to use that variable Now from this point forward to reference both the data. So there is a DateTime object, it's got code in it, it's got data in it and I've got $now that points to all that stuff. Okay? That's what happens. So I can sometimes just say give me a new thing or I can give it some parameters about what I want that to be so in this case, you go read the documentation, today plus one week, that's giving you a week from now and we get two variables, one, there's two and we have two DateTime objects now. Objects DateTime as the class but these objects and $now points to one of them and $nextweek points to another one and somehow, the data inside these is different. This is the now data for $now and there's now data for $nextweek. Not because any of the variable Next Week but because I told it to construct it in such a way that its initial values were next week. I'm going to say, Okay, hey, Now variable, format yourself. I'm calling the format method which is inside this and then I'm going to call $nextWeek format which is inside this. That's how you do it. There's this little arrow guy, this little arrow guy right here, says, Here's an object. Call the method name format within that object, so arrow is an operator. Method within or attribute within if there's data that we can access directly. And you pass into format the actual format that you want, kind of like this. You don't have to pass the actual object in because it's implicit because this format code runs within the object so it knows which object it is. So if the $now has a set of data and the $nextWeek has a set of data and it's slightly different, it knows when we call format, it's going to call this set of data, and if we call format here, it looks at that set of data. So that's how this comes out Now and this comes out Next Week because it is both code and data that's self-contained in two objects made from the DateTime template. Up next, we're going to make our own class and then make some objects with the class that we make ourselves.

Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?