# **Learning TensorFlow with JavaScript**

# 

### **Learning TensorFlow**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Emmanuel] Have you ever wanted to learn how to apply machine learning and deep learning to your projects, but been limited in your options, due to most libraries being on Python. Have you heard about this TensorFlow.js but don't know where to start? If you've answered yes to any of these questions, you've come to the right place. In this course, we'll take a look at TensorFlow.js and the basics, core concepts of machine learning with this library, all while applying the knowledge through examples. Hi, I'm Emmani Henri, and having worked with TensorFlow in Python, I was really happy to see this great library imported to JavaScript and able to show you how to work with machine learning. We'll first take a brief overview of what TensorFlow is and take a look at the few examples of its use. Then we'll go through TensorFlow basics, such as tensors, layers, and more. Next, we'll create our own first project from an example and start playing with datasets and models. Finally, we'll take a look at how to import Python-based models into TensorFlow.js. So, if you're ready to apply machine learning to JavaScript, open up your favorite editor and let's get started.

### **Course prerequisites**

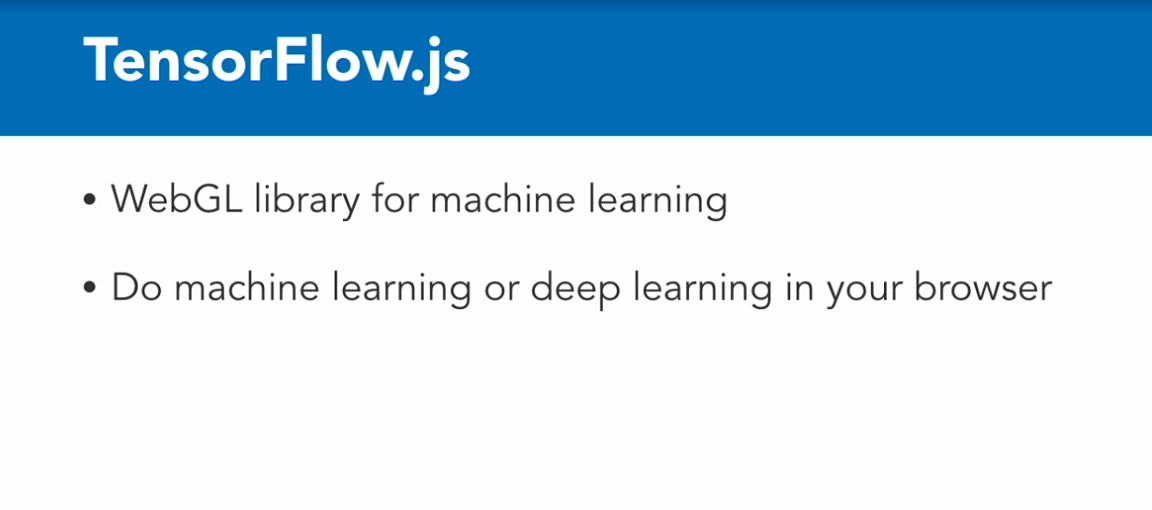
Selecting transcript lines in this section will navigate to timestamp in the video

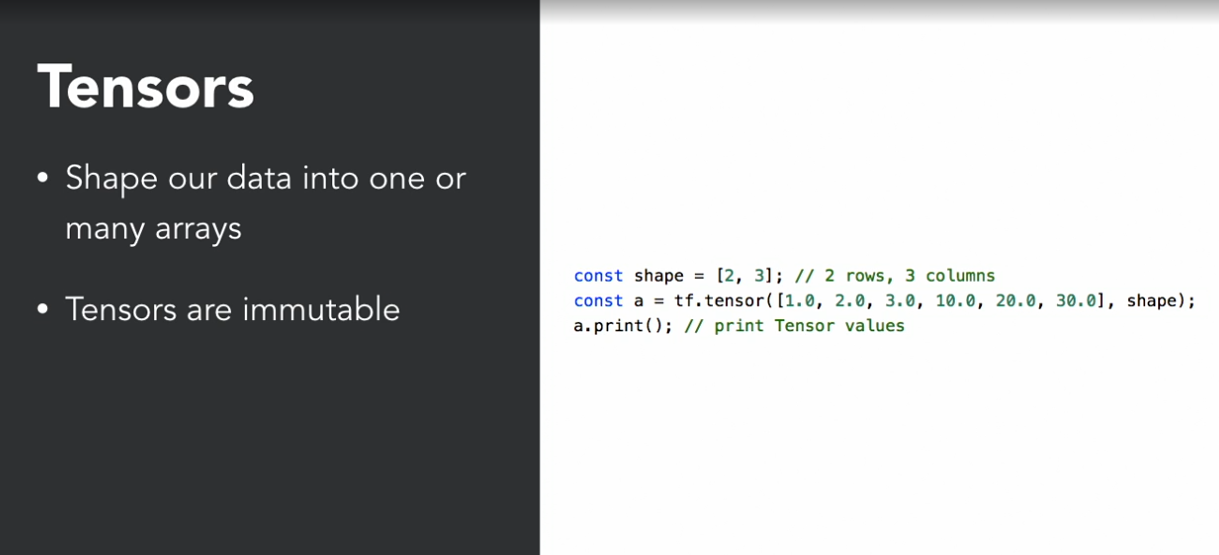
- [Instructor] Being entry level course for machine learning and deep learning with TensorFlow JS, now prior experience with this library is needed. Also, we'll briefly go through some machine learning concepts as we work with the library. Some knowledge with ML or a basic AI understanding isn't a must but definitely will help you. Please be aware that experience with JavaScript, especially ES6 syntax, is essential for this course. So if you need to brush up on your JavaScript, we have some great courses here in our library. In this course, I'll be using a Mac, but you can follow along on a PC as the tools work essentially the same on both platforms. Also, I'm using Visual Studio code as my code editor. It is free and simple to install, but feel free to use whatever IDE you prefer, although I do recommend using an editor with a built-in terminal

### **Introduction to TensorFlow**

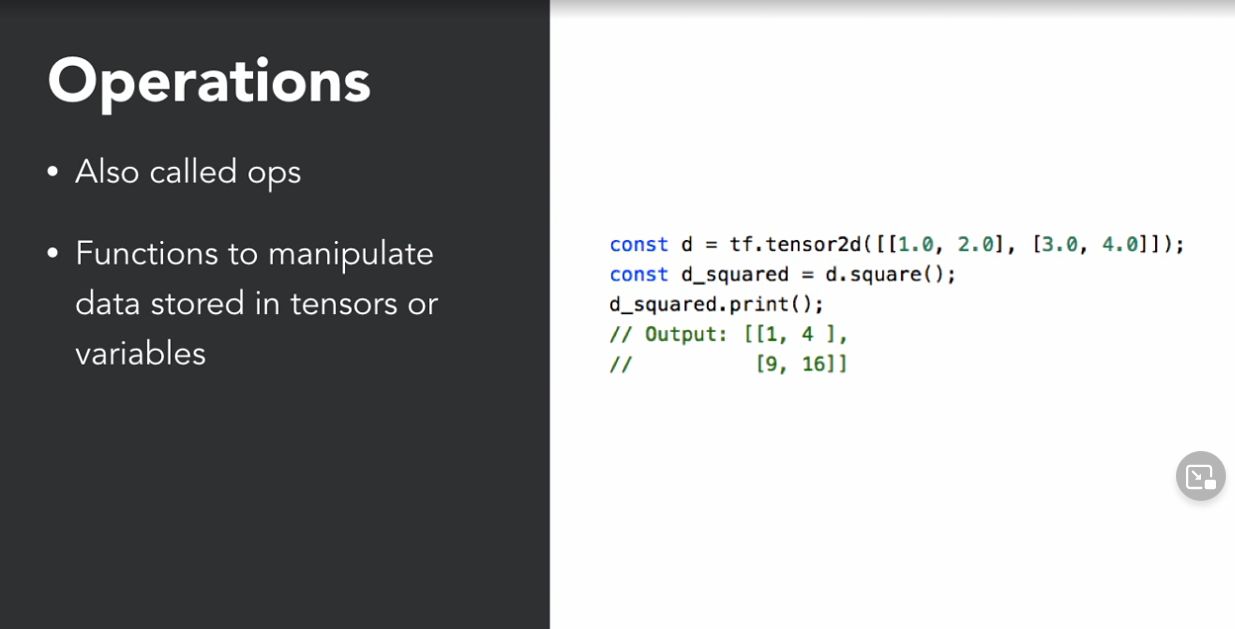
Selecting transcript lines in this section will navigate to timestamp in the video

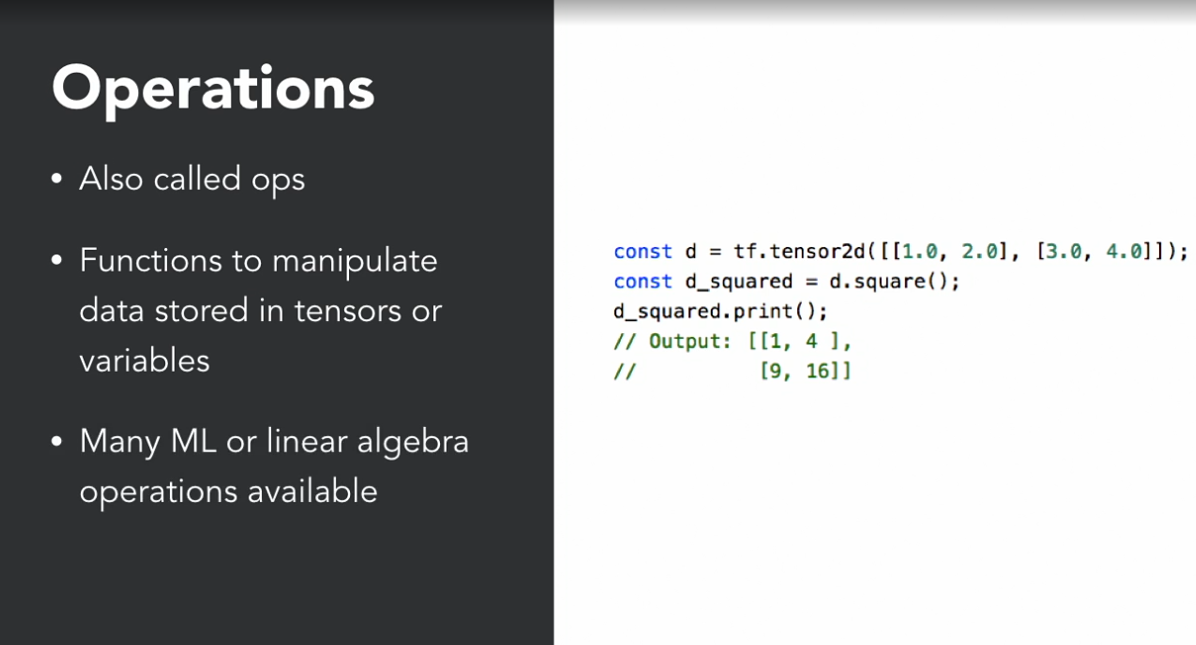
- [Narrator] If you've never heard about TensorFlow.js, it's basically an OpenSource WebGL JavaScript library for machine learning. It allows to train neural networks, therefore, also deep learning in your browser. It is based off the regular TensorFlow library built in Python, and borrows all the concepts form its parent library. Although covered in more details later in this course, we'll briefly take a look at the main components of this library. First, you have tensors, which is the central unit of numerical values where we shape our data into one or many arrays. Tensors are immutable, therefore, the values we assign to a tensor can't be changed. Then you have variables, which are a set of values that you can assign and can be changed. And if you need to set values that will be changed as you train your models, variables are best. Operations, or ops, are functions that allow you to manipulate data stored by tensors or variables. It provides all kinds of machine learning and linear algebra operations to explore your data. As mentioned before, if applied to tensors, you can't change this data. Models allow you to use operations to create an output. In other words, this is where you manage to come up with your machine learning or deep learning results. You grab the data off your tensors, and through a function get an output, which is the result of your machine learning or deep learning model. In this case, we output 24 with an input of two. So to summarize all these terms, you feed data to your tensors or variables, and then build a model with operations to come up with a machine learning or deep learning result. This is when it all comes together.

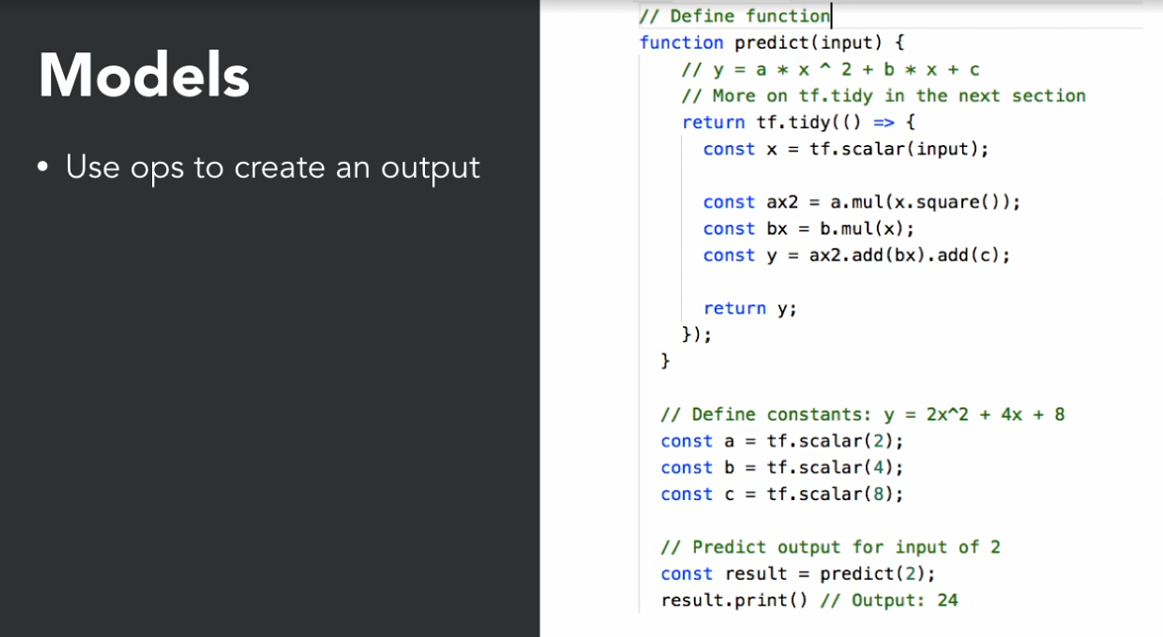








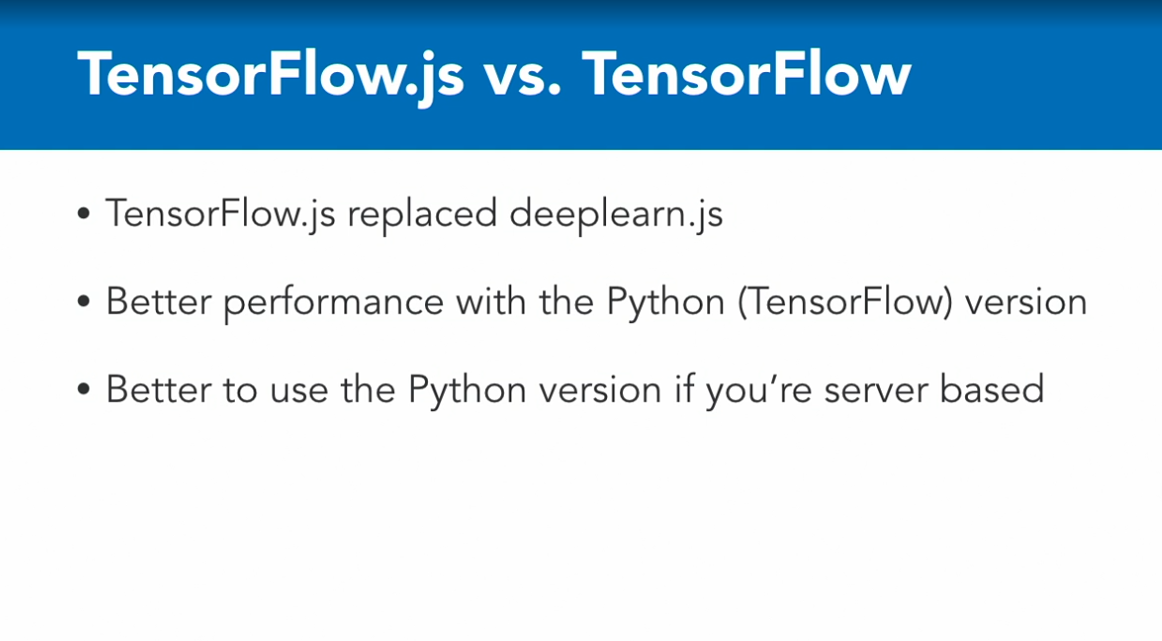




### **Differences between versions**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] If you ever used Deeplearn.js, TensorFlow.js has basically evolved from this library with the addition of several tools to import TensorFlow in Keras models and has now replaced it. What are the major differences in between the regular TensorFlow and the Java Script version? One, is performance. The regular version runs 1.5 faster than the Java Script version, and that is mostly due to how it is used at its core. TensorFlow.js runs in the browser, therefore, a lot CPU processes are used for your browsing experience, plus running it on WebGL versus Python with AVX. If you've ever used TensorFlow, the non-js version, in the past, you'll notice a performance decrease. Also with a recent wrapper, you can run TensorFlow.js in Node.js, so not in the browser, if you choose to do so. But in most cases, you really want to use the js version if you want to do Machine Learning in the browser. And for any other use where you want to play with data in the server environment, you'd use the regular version of TensorFlow. When you view this video, TensorFlow.js may have introduced new ways to use it, and if so, make your own judgment on which one to use.



### **Introduction to machine learning**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Machine learning is the ability to learn without being explicitly programmed. Well, what does that mean? In essence, computers or CPUs are pretty dumb if we don't program them. Without AI, a computer usually works in a simple input to output paradigm. We input a command either through programming or UI, open Word and then the computer responds with opening the program, the output. Another example is your calculator. We input two times five and the calculator outputs 10 as the response. So this simple input to output process is how computers used to work before AI started to permeate every piece of software. In a machine learning paradigm, there is another factor added to this input to output equation, the learning part. We have an input, learning model, and then the output. In this paradigm, the machine learns from your inputs and makes better output over time. Let me demonstrate with a simple Google search. If you were to type and make the mistake to type into Google search this query, what is the biggest dessert in the world, when you meant desert, in a simple input to output paradigm, Google may show you the biggest cookie or cake in the world. But because Google is built with AI at its core, it will have inferred that when someone is asking for the biggest dessert in the world, they probably are looking for a desert, not food. And this is where machine learning comes into play. Over years of gathering data or training the ML model, Google has learned that in most cases when someone is searching for biggest dessert in the world, they truly mean desert and that is the important item of machine learning. It needs to be trained for its model to be efficient, accurate or in other words intelligent. A machine learning paradigm needs to be fed with hundreds, thousands of more data sets to work. A great example is when DeepMind trained their machine learning computer to play Go, a Chinese board game. They spent hundreds of hours of feeding their machine all types of plays before the machine was able to predict what move to do next. So this is what machine learning is.

### **A TensorFlow demo**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So before we get started on building our own project, let's go through a demo of what TensorFlow could do. And if you want to test the same demo, you can go to js.tensorflow.org and then scroll all the way down to the Pac-Man version here. So click on Go to Demo, and it's gonna ask you to actually turn your web camera on. And as you can see right now, you can see me. And what I'm gonna do first is actually train this Pac-Man game or Pac-Man to actually follow my head. So whenever I turn right or I turn left or I turn up or down, Pac-Man is actually going to follow my movements and move in the game. And that's a good example of machine learning. So the first thing we need to do is kinda train or feed data into the model. So the first thing I'm gonna do is literally take pictures of my head turning right, left, up, or down and actually feed that into the model. So let's go ahead and do that. So I'm gonna turn right first. And then I'm gonna turn left, and then click to grab a sample. And then I'm gonna turn up and then down. And then let's do that again for left, right, up, and down. So right now, as you can see, I have six examples for the up, five for right, five for left, and five for down. And then what I'm gonna do is train the model to actually accept these values as turning right, left, up, or down by clicking on Train Model. So now it's going to train it, and then the loss means that if I am moving right and it's not reading am I right properly, it's not gonna move. So let's go ahead and play Pac-Man, and the only thing, I'm not gonna be able to see where Pac-Man is as I turn my head left and right, but you're going to see the result on the actual video, so at least you'll see that it works, not me. So I'm gonna play, and what I'm gonna do is right now turn left to make Pac-Man go there. Up, left, and I don't know where he is, so I'm gonna turn right, down, right. And then I'm gonna try up. I really don't know where he is. And I got eaten by the ghost. And then, and so on, so forth. As you can see, as I'm actually moving my head, the actual joystick is recognizing what movement I'm making and making Pac-Man move. Now the cool thing about all this, and let me just stop this. The cool thing about this is that if that wasn't accurate or if the movement was not always accurate, what we could do is train our model with more examples. So we could literally add more examples to these and train it to be more accurate. And this is what machine learning is about. By actually feeding the model with more data so it actually is more accurate and get better results over time. So hopefully this actually helped you understand a little bit more about machine learning and with some examples built entirely in TensorFlow.js.

### **Initial project creation with TensorFlow**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So let's get started setting up our project so we can explore TensorFlow.js. So the first thing I want you to do is open up Visual Studio Code if you don't have it open or use a different editor, feel free to open your editor. And what we're gonna do first is install and extension called Live Server inside of VSCode. So click on the extensions and look for Live Server and you can click on it. And basically what Live Server allows us to do is to create a server automatically with a simple HTML file that we are going to create. As opposed to create all the Note.js server code and all this, this creates it automatically for us. And once we have an HTML page, we can actually go and create the server. So let's install this and then reload VSCode. Once it is reload we're ready to get started. So the first thing we're gonna do is create a brand new folder that we'll call TensorFlow. And let's go to the desktop, create a new folder. I'm gonna call this TensorFlow and then simply drag and drop this folder inside of VSCode so we open that folder, like so. So now we have our folder here, we're ready to create new files inside of that folder. So what I'm gonna do first is create an index.js and in that index.js I'm gonna do a console log that will say: It works! And that's it for now, let's save that and then create a brand new file called index.html. And what I'm gonna do next is use ML to create a quick HTML document for me. So I'm gonna do doc and then hit tab and then it creates the base HTML document for us. And then what I'm gonna do is simply add the script here with the source of index.js, and then just change the title of the document to TensorFlow demo, perfect. And then what I'm gonna do to make sure that I have TensorFlow loaded into my program is add a script at the top here. So let's add a script and then do source and the official CND for TensorFlow.js is: https:// cdn.jsdelivr.net /npm/@tenstorflow/tfjs@0.12.0 so that's the current version, zero, like that. So if you want to make sure you get the right address or latest version you can go directly to TensorFlow.js website, tensorflow.org, and you wanna make sure you do the js.tensorflow.org and then getting started and then you choose to see the actual source here. So we could literally copy that as opposed to type it then paste it in here if you don't wanna make any typos and so on so forth. Alright so once this is save, we can actually start our Live Server. So the way to do this is by shift+command+p to bring up the command for VSCode and search for Live Server and then open up the Live Server. Once it is started you're gonna see it at the bottom of VSCode here and your page is going to automatically open like it just did. So I'm gonna bring up the developer tools by doing option+command+i and take a look at the console and just do a quick refresh because this is normal. And then it says: It works! If you see something that says WebGL, it's not supported on this device, this is okay for the actual course. You're still gonna be able to run the examples that we're going to do in this course. You're not gonna be able to access the GPU of your computer which means that when we're running TensorFlow, TensorFlow usually is a GPU-based processing library. What it's gonna do is use your GPU to run the examples or run the actual Machine Learning or Deep Learning layers and so on so forth, the models. But if your device does not support WebGL, it's okay. What is gonna happen it's still gonna run through the examples and use your CPU instead. So this is it to setting up TensorFlow. Let's move on.

**Live server install VS code –VVI**

### **Your first tensor**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] A tensor is the central unit of data where you can add values in multidimensional arrays. We can shape our tensor to be multiple arrays of a specific number of columns. Let's take a look at how this works. So, the first thing we are gonna do is remove the content log here, like so. And what we'll do first is create a variable called "data" and use dtf, so basically everything starts with tf. So when you're leveraging the library TensorFlow, you start with tf, and then pass whatever method you need to use. So for example, here we're going to create a tensor. And that's a method that takes the options of what are the tensors. So this is a method that actually takes the tensors, or the array of numbers. So, in this case we'll create four arrays of two numbers. So, let's go and create that, and let's create the first array. And add "4,6" in this first one, and then create a second one, and in this one lets do "5,9", and then let's create a third one, and in this one we'll do "13,25", and then in the last one, we'll do "1,57", I'm just doing random numbers here. Alright, so we created our first tensor, so now let's go ahead and print that one. So the way you print a tensor, you basically do "data.print" and that is a function, and then just save. And as we save our live server is going to automatically reload our server. So let's go back to our page here and we see our first tensor showing up. So, a tensor of four arrays, with two data point inside of each. So, this is how you create a tensor. So, this is how you would create a tensor manually, or it would infer that you want a tensor of four arrays with two data point inside, but you could also shape it before. So, you could do something like a shape. So, you could do a variable called "shape", and that shape would be "4,2". So, "4,2" which means four rows of two columns. And then instead of actually doing this manually, you would just remove all of this, and add all your numbers, and then add the shape as an argument, and it would still create the same tensor for you. So, let's go ahead and save this. And let's go back to our browser, and we got the same tensor, with four arrays of two columns. So, this is pretty much how you create a first tensor. So, let's go and then save that, and let's move on to the next video.

### **Tensors and variables**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] As mentioned before, tensors are immutable, meaning that once you set the values, you can't change them. So what are variables? They're pretty much the same and are set exactly the same ways as tensors. The only difference is variable can be mutated, or changed. So let's explore this in code. So the first thing I'm gonna do is just move my data print just a tad below, like that. And what I'm gonna do here is set up a new variable. So let's go and set variables and what we're gonna do is use the zeros method that's available inside of TensorFlow to create a series of zeros. With zeros method. So let's go ahead and do a const and we're gonna call this second tensor data2 so we use tf and now instead of tensor we're gonna use variable. And inside a variable, we're gonna set zeros and we're gonna do eight of them. So that's how you set a new data set with variable with eight zeros inside. Let's make sure I don't have any typos here so we're all set. So now that I have a series of zeros inside of a variable, if that was a tensor, I wouldn't be able to change those values. So let's print data2 first. Let's go to our browser and we see a series of eight zeros. So if that was at tensor we couldn't change it but because it's a variable we can actually mutate or change that data. So let's go ahead and do that. And what I'm gonna do is keep that print here and then do a second print after we change the data. So let's take data2 and assign. So this is how you actually change the data so you use the method assign and then assign a set of values. So what we're gonna do is use tf.tensor1d so I'm going to introduce something new here. Tensor1d means that we're doing values with only on dimension of an array. If I would do two Ds that mean I would have two rows of array and so on so forth. So that goes up to 5d, and if you wanna do something that's only a number or just one value, it's scalar, like that. So in this case we're going to do tensor1d, which is a method, and then enter a whole bunch of numbers. So I'm just gonna do random numbers. 12, four, five, six, 56, three and then 45. And then I'm just gonna make sure that this is done in a clean manner as well, perfect. Now I've assigned new values and I have eight. One, two, three, four, five, six, seven. No I don't have eight. And then three, now I have eight values. Now I've assigned these values to the variables in the data2 set that I initially done. And if I had done a 2d-- oh I changed this for s, it's a d. So if I had done something like a 2d and then shaped it, my problem would be that it's not in the same shape as my original data. So we wanna make sure that whenever you assign new values to your data sets here, you wanna make sure that it's in the same shape. So one row of eight columns. Otherwise you're gonna have some issues. Then what we're gonna do is print it again, data2.print. Alright so let's save that. Let's go back to our browser and now we have the initial values at zero and then we've assigned new values for 12, five, six, 56, three, 45, and three, and we got this brand new value assigned. Let me show you what I meant by not having the same shape. If I did only seven numbers as opposed to eight, as the original shape, and saved it, then I would have an issue here. So shape of the new value: seven. And previous value must match and so on so forth. So let's just undo that, save again, and let me just add a little note here. This is where we assign new values with one dimension, and save that. And, let's move on.

### **Operations, or ops**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Operations, or ops, are the basic operations you can use to manipulate, analyze, infer, or do all sorts of operations on the data. They are essential for machine learning. There are dozens of operations we can use to manipulate data inside of TensorFlow. You have math operations, image operations, linear algebra, and many more. Let's take a look at a few of them. So before we actually do anything in VS Code, let's go to the browser for a few seconds. So if you're not on js.tensorflow.org, go to that site and then click on API References. And then what you need to do is scroll down until you see Operations. It's right after Layers. And here you're gonna see the operations available to us. So there's arithmetic, basic math, convolution, reduction, and these are all normal to people that actually do machine learning. If some of them don't make sense, it's okay. So these are basically machine learning-related, so you can literally read about RNNs and it's gonna make more sense afterwards. So let's go and try one in our code. So we're gonna do basically the layers, so if you wanna have more information about it, you can click on it and see the syntax. And in this case here, we're going to add two tensors and basically print out the results. So let's go into VS Code here, and the first thing that I realize here, I can use the data and the data2 to do basically an addition because they're not in the same shape. Although I have eight numbers here and I do have the same eight numbers here, this is in 1D, so one dimension, or one row, as opposed to this one has four rows of two columns. So they're literally not in the same shape. I couldn't add those two together. So we're gonna have to create a brand-new data set to actually do the addition. So what I'm gonna do is simply go down here and create two new data sets. So let's go ahead and do const data3, and let's create a tensor, a one-dimensional tensor, so we'll do 1D, and let's add four numbers. So let's do four, six, five, and then nine, and let's just copy this line and paste it on line 17 and then create data4, and this one let's just enter a bunch of other numbers. You can enter anything you want. Awesome. So now let's print those two before we actually add them. So let's print again. So copy line 14, paste it here, and do three. Let's paste again with Command + V and do four. And then finally let's add them. So what we're gonna do is do data three and use the add function. So again if we go back to the syntax, this is how it works. So we select the first tensor we want to add and then do the add method and then pass the second tensor that we want to add. So let's pass data four. And then what we could do is also print right away. So this is a nice way of doing an addition and then printing it right after. And then let's copy line 22 and let's do a second operation, something else. So let's multiply instead. So if I remember correctly, multiply is mul. So let's go back here and let's take a look at mul. So, yeah, mul multiplies. Or you can try anything else if you want as long as you're understanding what you're doing and you understand the result that's gonna come out of it. All right, so let's multiply and then print it. All right, so let me add a little comment before we actually go to the browser. Adds and multiplies and prints. And then here prints. Finally, here we're creating two new tensors. And let's be specific, one-dimensional tensors. So I want you to be comfortable with talking about dimensions, because in machine learning and deep learning, we're always talking about layers, we're talking about dimensions, talking about inputs, the number of units, so all these new terms that you may not be familiar with, you want to talk about them. You want to get your mind around these different terms so when somebody talks about machine learning, you'll know what they're talking about when they're talking about layers and models and tensors and all these different things as related to TensorFlow. All right, so let's save that, and let's go back to our browser and look at the results. All right, so first we did print the first two here. So this is from the past exercises. Here we're printing the first tensor, and then the second one, and then in this one, we're actually adding them. So four plus five equals nine and so on, so forth for each line. But then we're doing the same but multiplying instead of adding. So four times five, 20, six times four, 24, and so on, so forth. Five times 23 equals 115, and so on, so forth. So this is a nice introduction to operations, and you'll see them everywhere when we start building our models. Let's move on.

### **Model introduction**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] The model is where everything comes together. This is where we provide the input and the model generates the output. It can be a simple function or it can be made of multiple layers, which is how machine learning works. We'll explore how to build a model with layers in another video. And in this one, we'll build a simple model based on what we've done so far. So what I suggest you do here is either comment all this code here so you can keep what we've done so far. But in my case, I'm going to delete all this code and start from scratch, so I can explain everything as I code. So I'm gonna highlight every line and then start from scratch. So what I'm gonna do first is define my model. So model is a pure function, so let's go ahead and start a function that will be called simpleAdd. And that function will take two inputs, so input1 and input2. And then we'll open up our curly braces and then we'll return from tf, tidy. And I am going to explain what that is in a second. So tidy is basically the tool within TensorFlow that manages the memory allocation inside of your GPU. Because when we're using WebGL, we are using GPU cycles, we need to use tidy, which basically frees up the GPU memory once it is done. So when we return what is inside of tidy, it will automatically clean up the memory inside of the GPU or anything that we're using from TensorFlow and free that up, so this is why we are using tidy. So let me just quickly add a note here. Tidy is used to free up GPU memory once the function returns. All right, so inside of tidy, we need to basically run a function, so we'll do that. And I'm using ES6 syntax here. So what we gotta do first is create one variable, which is the x1. So if you've followed any of the courses on machine learning or if you're familiar with features, this is where we're going to start using the x as the features in our model. So let's do input1. So we're assigning the value that's coming here and then we're going to create an x2, which is coming from input2. And then the result is y in machine learning. And we'll basically do x1 and add x2. We're doing an addition again, so we're using an operation inside of the model. And then finally, we'll return y. Perfect, so now that we have a model defined, let's go ahead and use it. So what we're gonna do is create a set of data, so let's do data1 and create a one-dimensional, so tensor1d. So let's do a one-dimensional array, so we'll do four. Again, we'll use similar numbers that we've used before. So five and then nine, so you can understand that the model works exactly the same way. And then let's do a copy of that line. Oops. Undo. Command-C to copy and then Command-V to paste. Then let's do two and let's do five, four, 34, and then let's do 21. Okay, so we're basically here creating the new one-dimensional tensors or arrays. And then we'll use the model. So we'll create a new variable, which is called data3 or maybe, we'll call this result instead, which uses the model, so simpleAdd and then passes the two one-dimensional arrays or tensors that we've created as the value points. And then let's just do a print. Result.print. Perfect. So using the model to input to output. And then we're printing. Printing result. Okay, so let's save that and let's go to our browser and now, we got the result. So if we look at what we've done here, we did an addition. So four plus five equals nine, and so on and so forth. So the second row was four plus six and then nine plus 21 equals 30 and that's what we get. So that's pretty much how a simple model works. So you get inputs, you get some operations, and then you get an output.

### **Layers introduction**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] When you start doing deep learning, the concept of layers becomes very important. And in TensorFlow.js, you can build your model out of layers to construct your type of neural network. Layers are the building blocks and basically performs operations with its inputs and then output the data to be ready for another layer or final output. Let's explore the API docs to see which ones are available and then build a simple sequential layer. If you go back to the website, to js.tensorflow.org, and you are in the API references, once you get there, scroll until you see LAYERS. It should be roughly after the models and you'll see all the layers that exist. You have the basic layers where you can do activation, dense, dropout, embedding. You have the events activation when you can do an elu. You also have the convolutional layers, the merge, normalization, pooling, and so on, so forth. Basically the layers pulls from the output of the previous layer and then does some computation and then outputs to the next layer until you get through the entire network. So this is how it works. Let's actually do an example in code. Let's move to VS Code, and what I'm gonna do is remove that code here. If you want to keep it, feel free to comment that code and then type after. Otherwise you can also take a look at the exercise files if you wanna bring back some of the stuff and the comments that I've done. I'm going to erase all this and I'm gonna start by creating a variable called model. Let's just add some comments here. This is a sequential model that we'll create. Basically in the sequential model, the outputs of one layer are the outputs of the next layer. Very simple, not too complicated. So sequential model. Let's go ahead and create it. We use, again, the tf, and do sequential, which is a function. And now this variable is a sequential model. What we'll do now is add a layer. The first layer you add to an actual model, you need to define what is the input shape. The input shape is the actual number of rows and columns in a specific layer. All the next layers that we're gonna see after are going to inherit from that first layer. Let's go ahead and do that and do tf.layers. And for the layers, we're gonna do a simpleRNN. The first thing we need to define is the input shape. So again, this is only needed on the first layer. Let's do an inputShape. For example, I do something like 20 by 4, so this is the shape of my layer. Then what we're gonna do is, in this layer, how many neurons we're gonna have. This is what is called units. This is the number of units or neurons. We're gonna call this units. Let's say, for example, we do 20. And then what is the weight of that layer? That layer will compute the data to spit it out to another layer in our neural network. So the weight of that particular layer has an impact on how it will actually impact the final result. What we wanna do is add a weight to it. I'm not gonna get into the details of what this particular weight is and what are the others, but if you wanna have more details around the weight itself, you can go the documentation for TensorFlow. This is called recurrentInitializer, and for this one we'll do a 'GlorotNormal'. That's pretty much how you would set up the first layer. In a neural network, you usually have more than one layers, and each layers with, in this case, sequentially output data to the input of the other. So if we wanted to add multiple layers, all we had to do is do another one model.add, and then add a layer. This could be a number of layer; doesn't need to be a simpleRNN again. If you go back to the layers documentation, you could do any of these layers and actually impact the final result based off the layers you put in there. This is how layers work inside of TensorFlow and how you add them to the model. Let's move on.

### **Import example project**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's import a good example project so we can explore further models, layers, and see how training works in Tensor Flow js. So, the first thing you need to do, go to your browser to github.com/tensorflow/tfjs-examples. You want to get to the site, either Clone or download this particular repo, and I'm going to download it. Once you have it on your system, extract the zip file, or grab the documents here. And, once you have it open, what I want you grab is the mnist example, here. So, this project here. So, what I'm gonna do is drag and drop it to my desktop. You can put it anywhere you want, this is where I'm gonna put it. And then, what I want is to open this inside of VS Code. So, I'm going to go back to VS Code, and make sure that I don't have anything else open. So, if I click Explorer, here, I don't have any projects. If you do have something, click on file and then close folder. And, let's go and open folder, and then select mnist from the desktop. So, if you wanna run this example in your browser directly from the files, here, you can do that by installing with the yarn or npm install. So, let me show you how that works. So, go to view, integrate a terminal, and then you can do yarn to install, or npm install, if you don't have yarn installed, and it's going to install all the dependencies that are in the package.json file. So, I'm not gonna show the example of this particular project. I have a nice one on the web. And, I'm going to show that later on. So, let's go ahead and explore layers, models, and a whole bunch of other things in this project, in the following videos.

### **Exploration of the models and layers**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Now, let's continue our expiration and take a look at the models and layers we've done in this project. So now that we have the data inside of our index.js here, remember the data at this point is transformed into actual numbers. So the images are not images. They're actually numbers. So once we have the numbers for the pixels of these images, we pass them into the model. So we can actually determine based on the patterns of these pixels, what numbers they are. So first, we create the model here, which is a sequential, which means that one row of layer comes after another. So the input of a layer is the output of the previous layers, and so on, and so forth 'til the end. And then we add a layer into our model. This is a convolutional neural network layer. And then we have the options based on the layers. So these are the options based on what we wanna do with the image. So if you wanna have more information about why they selected the convolutional layers, and why they used a RELU activation inside of that layer, you can read the comments here or read more on actual machine learning terms. And then we add more layers. So we have a maxPooling layer here. So basically the output of this layer will be the input of this layer, and then so on so forth. So we have a second convolution layer here, and then maxPooling again, and then we flatten the layers so we can get to an actual result at the end. So this is pretty much how any models work. So you have the model at the top, and you have multiple layers that actually computes that data to get to a result. And then we'll need to train our model to make sure that we have the proper results. And we'll take a look at that in the next video.

### **Exploration of training the model**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now let's take a look at how the model is training into a project and the options. So let's get back to index.js and once you get back to index.js, take a look at line 94 inside of that file. So basically this is where we set the options to train our model. So what we need to first is define the learning rates. So this is something that you need to test, usually a .15 would work but based on the type of dataset that you're training you may need to revise this so, sometimes you need to go higher or lower, take a look at what the test results are and then find your own number. Then you pass that learning rate to a variable optimizer where you use the stochastic gradient descent, that's the usual, basically, algorithm that we use. There's a few others that you could use but this I primarily the one that most people use. Again, you can take a look at the comments and why this is the one that most people use. And then you actually to the model you compile and train with the options here. So this is the loss algorithm and the metrics that we're gonna use, you can pass, in most cases, use these here to just test it and then try some other options based off the documentation. So the batch size, the train batches, the test batch size and the test iteration frequency are all part of what is the volume of your test, so the batch size is basically how many items are in your batch of tests and then train batches are basically the number of tests you wanna train before you can log in and say your model has been trained. So once you have all these you can run the async function and train your model with all these options here. So this is the function that will actually train your data with all the options that we just set and eventually give you the results that you need. So this is an overview of what functions and what options that you need to set inside of your index.js file in order to train your dataset and your models to actually get some results. I would strongly suggest that you take a look at all the other examples including this one in more details and tested even further than what we're gonna do in this course because this is an introduction to TensorFlow.js but you need to learn a lot more about machine learning, deep learning all these different terms that are part of that world in order to understand everything that's in here but at least it gives you enough to get started and then move onto more information around these subjects.

### **See the live example**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now let's take a look at what the project looks like. It's very similar to the Pac-Man game that I showed earlier, it actually needs to be trained. It actually needs to load this data set and actually train itself before it can give you results. You can either run the MNIST project here or you can go to the official example on the web, and that's what I'm gonna do. Open up your browser and then go to storage.googleapis.com/tfjs-examples and it's actually this address here, so examples/mnist/dist/index.html. Once you get to this page, you're gonna see something like this happening. Right now what it's doing, it's loading the data and then it's training the model to make sure that we have something that is as accurate as possible. While it's doing that, you're actually seeing the loss of the training and the accuracy of the training. So we're at 98.44%, which is pretty good and we have a last loss of 0.21 in the training set. That's pretty good, so that means from all the sprites that were sent to the actual model, of all these sprites we have only about 1% point 56 that is wrong and it represents what we're seeing here. So this has been fed all these numbers that were in the sprites and this is what the system has predicted. So this has predicted that this is a three. This is good, two, one, seven, and so on so forth. The two losses are this one here, so zero as opposed to five. This predicted that this was a six, that looks more like a nine, it could be also a four depending on your handwriting. And these eight instead of nine and three and so on so forth. So this is an example of the actual project that we just taken a look at. Feel free to take a look at that project in more depth. Read the comments inside of the project. Google was very good at actually adding a lot more details as to what each line was for and actually introducing you to some of the terms. In most cases, you're going to at least have a good understanding of what these machine learning terms are if you're not familiar with them. So you need to have a basis of machine learning before you actually understand all the options that are inside of the library in TensorFlow. If you're missing some of that, there's a couple courses in our library that you can take that will give you some of that information. If you want to also extend that knowledge further, there's also a Machine Learning Crash Course on Google's website, so all you have to do is machine learning crash course inside of a Google search and then click on this one here, so Machine Learning Crash Course on the Google's website. This is a solid course on all the terms, all the stuff that you need to know about machine learning, about neural networks. It's gonna give you enough to be dangerous and they're using a lot of TensorFlow inside of that crash course, so you'll be even further introduced to these terms in a practice.

### **Use Python-based models in JS**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] There are many other examples with code you can further explore to get a good grasp of how to use TensorFlow.js, so let's take a look at a few. I always say that the best way to learn is to take a look at the examples of what others have done and then try to reproduce it with your own experience. So if you take a look at the main website here, so js.tensorflow.org and you scroll all the way down here where we show the demos initially, each of those demos have actually code that you can take a look at. So if you click on here, you're gonna get to the code related to the Emoji Scavenger Hunt. So you can clone this code and actually take a look at how they built it. So you can even do it from here. So if we take a look at the source and take a look at js files, and then go to the index, and then scroll all the way down here. You can take a look at, well this is just the UI stuff, but we can go back here and I believe somewhere in there there's the mobilenet. So that's the actual image code here that we have and then you can take a look at either the classes probably have everything else, so yeah. So we have the functions here to actually run a couple of these things. And eventually you have the dataset which is fed by you. So if you wanna take a look at any of the examples here, they each have code associated to it. So the teachable machine, you have code here, same thing for this one, and so on so forth. So you even have a baseball example where it actually determines based off what the pitcher has thrown. Is this a fastball, a changeup, or a curveball? I'm not familiar with a changeup type of throw, but I'm assuming this is related to the yellow line here. So you can actually take a look at the code and where the data has been pulled off is the actual Baseball Association and you get some data to play with. So if you want to have even more examples, you can take a look at the GITHUB. You can take a look at the repo where you have all the examples. This is the actual library for TensorFlow, but the one with the example is github.com/tensorflow/tfjs-examples and you have a whole lot more examples here that you can play with and actually take a look at how they did it, and then do your own from there. So if you wanna build something that's similar to what are the examples here, you can at least have a good look at how they did it and get your project started from there.

### **Convert SavedModel to web**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now let's say you're coming from TensorFlow, the regular version, and you want to use TensorFlow.js. So, there's a way to actually import your Keras models or your TensorFlow models into TensorFlow.js and let me show that to you. So, the first thing you need to do is if you want to have more information about the tfjs converter, you can go to github dot com forward slash TensorFlow forward slash tfjs dash converter. And you're going to get more information about what that converter is, how does it work. What I'm going to show you very quickly how would that work. So, go to your terminal. And the first thing you will need to do is do a pip install tensor flow js and that will install the proper packages related to the actual converter. Okay, so once you have that package installed, you can go back quickly to the actual reference here and take a look at what is the actual command. So, that's very, very simple. You will need to grab the actual path of where the file is, so the model from Keras or from TensorFlow. So, basically you could copy and paste these examples here but the simplest way is this way. Let me just go back to the terminal and clear that up here. And go to this guy here. So, basically the path of this guy here. So, path, I don't know why it's all spaced up like that, but you would need to do tensor flow js underscore converter dash dash input format and then mention what is the format. So, if it's a TensorFlow format, you write TensorFlow. If it's a Keras format, you write Keras. And then the path to the actual file that is a Keras format and then the path of the files where you want it to end. And that's pretty much how you would do it. Once you have converted that file, you can go back to an actual document here inside of your js project and then import TensorFlow and then basically import your model. So, const model and then load where the model is. So, basically right now this is on a specific address but it could be anywhere in your system and that would be where you put the url of the model you are trying to import into your JavaScript system. So, that's pretty much how it works. So, if you're coming from TensorFlow or Keras and you want to try TensorFlow.js, that's how you would convert your model and then import it into your system.

**Tf-js converter**

pip install tensorflowjs

### **Next steps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] This was a brief and basic introduction to TensorFlow.js. Needless to say, there is a lot you can do from here. There are many tutorials and code samples on the TensorFlow.js docs that will get you going now that you understand the basics. You could continue working with this great library by applying it to your own project. If you already have an application built in Node.js and would like to start feeding the data from this project to TensorFlow, so you can enjoy the benefit of machine learning, this is the next step. Also, if working with TensorFlow really got you excited but found the JavaScript version to be a bit limited, you could get started with the full Python-based library. In this case, if you don't have any prior Python knowledge, take a look at all our courses on learning this language. Then, when you're ready, you can explore the full TensorFlow course we have in our library. Finally, if you got lost somewhere in this course due to your lack of Node.js or JavaScript knowledge, feel free to explore our dozens of courses on JavaScript, Node, Express, or any of the front end frameworks such as React or Angular to get you more development experience. Thanks for taking my course and I'll see you in a bit.