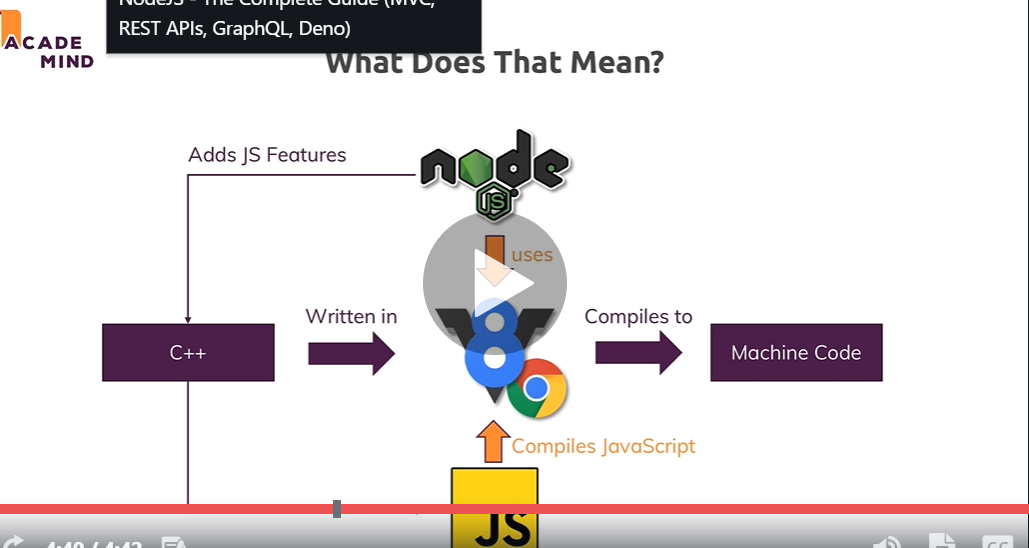
**Lecture 02 - What is Nodejs?**

So let's dive into the most important question, what is nodejs? Nodejs is a javascript runtime and now what does this mean? You know javascript, it's a programming language you typically use in the browser to manipulate your dom, to manipulate the page which was loaded in the browser, for example to open a popup, a modal or add any kinds of effects because Javascript is a language that runs in the browser that allows you to interact with the page after it was loaded and it therefore is a crucial part when it comes to building interactive user interfaces in the browser, so whatever your users see. However javascript is not limited to that. Nodejs is a different version of javascript you could say, it is basically built on javascript, it adds some features to it, is not capable of doing some other things you can do with javascript in the browser, so it basically takes javascript and puts it into a different environment. It allows you to run javascript code on the server you could say, in theory not just on the server but on any machine though. So it basically allows you to run javascript not just in the browser but anywhere else like a normal programming language, like normal programs on your computer or some computer in the Internet effectively making it a great choice for building web applications that run on servers which are just computers running somewhere in the Internet. So in detail, this means that we can use nodejs to run javascript outside of the browser, that is the core takeaway, now how does this work technically? Well nodejs uses v8 and v8 simply is the name of the javascript engine built by Google that runs javascript in the browser, so back to the browser we now are. V8 is simply the name the creators gave their engine and what does an engine mean? Well it simply means that engine takes javascript code, the code running in your browser then or in node's case which builds up on v8, also the nodejs javascript code, it takes that javascript code and compiles it to machine code and this is what your browser does too, what v8 does in your browser. It does take your javascript code and compile it to machine code because that is the code that runs ultimately on your computer and that can be handled efficiently. Now this is done by v8, v8 itself is written in C++ but that doesn't really matter too much for you, you don't need to write any C++ code to use javascript or nodejs but nodejs basically takes that v8 codebase which is written in C++ and adds certain features like for example working with your local file system, opening files, reading files, deleting files, these are all things which are not possible in the browser, you can't access your local filesystem in the browser for security reasons, so this is not supported. Nodejs adds these features to v8's engine you could say so that you can suddenly do that. Now nodejs does not run in the browser so these restrictions still apply, there you use vanilla v8, so v8 only without the nodejs extensions but if you then install nodejs, you can use it to basically use that extended v8 version to run javascript scripts on your computer which then suddenly can access these new features because they don't run in the browser but are directly executed through that nodejs runtime you could say. So this is how that works together and what nodejs does. It allows you to run javascript on your computer and it adds useful functionalities to the javascript engine so that you can do more useful stuff there than you can do with browser side javascript. Now one important note maybe on this point also is that of course some features are also taken away. In the browser you use javascript to interact with the document object model, so with the html elements on your page, if you just execute a javascript file directly, you of course have no attached page and therefore these features are missing. But this is a lot of theory, why don't we just have a look at this and see how we can install and use nodejs and what it actually does for us. Let's do that in the next lecture.



**Lecture - 04**

**Installing Nodejs and Creating First App**

So we want to get started with note, Jess, and for that we, first of all, need to install it to do this visit note Jess, Doug and then dear, simply download this latest version. In my case, that's 14. 11. It will change over time. When you're watching this, you might see a higher version. It is important to note that these versions will change frequently, but they don't really bring many new changes. It's most of the time just behind the scenes optimizations and bug fixes. I will keep those cords updated, though, as I have in the past. But with that, let's install this latest version for this simply downloaded onto your system. And I'm doing this on a Mac here on Windows. It's exactly the same. You can download the installer here and then you simply walk through that installer. And again, here, that is the Mac installer on the windows, you have a similar installer and you simply walk through all the steps you have there, leaving all the default settings which are set up for you. You don't need to tweak anything, dear. Ultimately, this will then install note, James, on your system, this JavaScript runtime discourse is all about and we need no to just to be installed in order to execute JavaScript code with no tax, which, of course, is our goal in this course. So once this is installed, we can use it and the first and quickest way of getting started with it and of using it is that you open up your default system terminal or a command prompt on windows. So again, here I am on Mac OS, so I'm using the built in terminal on windows. You would simply open up the command prompt or partial and you can check whatever node has been installed successfully by running node Dash V to check the wording of node jested was installed and there you should see the version you just downloaded and installed. And with that, if that succeeds, if you don't get an error, we know that it worked. Now we can use node and one way of using it is to enter an interactive mode node. Jass offers you the so-called Reppel, to which I will come back later. You enter this interactive mode by simply executing node as a command. Now, dissenters, a new mode in your terminal or a command prompt. And here you can run certain node commands. You can use it as a basic calculator to do basic math there. And you could do more than that. You could run some JavaScript code in there again. I will come back to this Reppel later. But actually Dissinger active mode might be nice to play around with Node. You're not going to use it to write real node programs for data. Instead, you will use a code editor and store your code in files and then execute these files with node. So let's do that now for Dad. Let's quit this interactive mode by pressing control see twice or control D or Taiping exit as you see here. And once you closed that mode, let's actually create a new project, a new folder in which we write our first code. And I did just that, I created a new folder somewhere on my system, and I opened it with my favorite code editor, visual studio code, which you can find if you simply Google for Voice code or you visit code dot visual studio, dot com visual studio code is a free code editor, which is great for web development and it's available for Mac OS, Windows and Linux. So you can use it on any operating system simply downloaded from code dot visual studio dot com and then walk through the installer that gives you to install this ed on your system. As a side note, you can use any idea and code editor of your choice. Of course, Visual Studio Code is just what I will be using throughout this course and what I recommend that you use if you don't have a favorite. Ed, of course, with it installed, you can also open your project folder, which at the moment should be empty with this tool simply by going to file and then open and then pick a folder in which you want to write your code files. Now, if you want to make sure that you've got the same look and feel as I do, you can go to view appearance and there you can control whether or not you see the sidebar, this activity bar on the left and so on. So that's how you can customize this. And in addition, under preferences color theme, I am using the dark plus theme and you can switch to this as well if you want to have the same colors. Last but not least under view extensions, you can search for the material icon theme, which is 100 percent optional, but which will give you a specific I can look, which you will see for all the discourse in my project. So that's why I will enable this icon theme here. For me, using it is optional, though. And with all of that, we got Decode editors set up, which we are going to use throughout this course or which I am going to use. And now we can write our first note code in a file. For that, I'll add a new file in there, which I'll name first dash app Dot James Dot James, because of course it will hold some JavaScript code in there. We can now write JavaScript code, which can be executed by Node JS and a very simple first code snippet we could write. Here is a simple console log statement which logs something to the console where we say hello from Node James, for example. This is code which would run in the browser, it will also run if we executed with no just now two executed with no charges. We need to execute this file here with no jass. And for that, the easiest way of doing that in pseudocode is that we go to terminal new terminal. This opens your default system terminal or a command prompt here, integrated in this idea and already navigated into this project folder. And then we can run this first abcess file by running node and then first the dash abcess. So simply adds the filename after node and then you will not enter this interactive mode, but instead execute this code, file this JavaScript file with no charges. And therefore you should see hello from Node Jacir. Now, that was a nice first example, but we can do more with Noad and to right and at least a little bit more realistic or more fancy application, we can also try writing some output's to a file instead of the console. And for this, we'll leverage one of the built in functionalities no trace offers. And that would be the filesystem functionality which enables us to work with the file system. For this, we first of all have to imported into that file to let Node know that we want to use this functionality. And the syntax for that is that we call require function made available by no notebooks. And we want to require DFS module, which is one of node's core modules shipping together with no notebooks. And I will come back to this import syntax and to the node core modules in greater detail later. For now, we just call this and restore the imported filesystem functionality in a simple constant here. And then we can use this file system to call right file sync, which is a method made available by this filesystem object which we're importing. And this method here will write a file to our hard drive. And the argument it wants is the path to the file, including the filename. And here we could named as hello to you. And then the second argument is the content of that file. And here we could store hello from Node G.S. again. So now this is our code here I am writing to a file by leveraging the filesystem module offered by NASA, and if we now save this file and then run Noad first abcess again, we should find a haloed text file next to our script file, which contains this content. And that is how we can use notorious. Now, obviously, we're just scratching the surface at the moment, we just learned about a brand new syntax, but you don't know from the browser and therefore we are going to dive way deeper into all of that over the next lectures and throughout this entire course.

**Lecture - 05**

**Understanding the Role and Usage of Node.js**

So you hopefully got a first picture of what nodejs is and why you use it and what you use it for. Now typically and also what we will do in this course, you use nodejs to run it on a server to write server side code and for that, we have to have a look at the full picture. We have our users using a client computer with a browser, their mobile phone with a browser, even mobile apps and we will come back to how nodejs can communicate with these later in the course too but for now let's stick to the traditional browser picture. We get our users using the browser and there we can use html, css and javascript, javascript in the browser to create webpages, right. Now they visit a page, mypage.com and they send a request to do so, for example by entering a url in the browser, a request is sent to that url. Now there, this server comes into play. We got our server, some computer running in the Internet which has the IP associated with that domain which is automatically resolved for us and on that server, we then want to execute some code that does something with the incoming request and returns a response, often but not necessarily always as you will learn, this response is a html page which the browser then in turn can display. By the way, it is not necessarily just html, it's also things like css files or javascript files with javascript code for the browser, not for the server. Now on the server, we typically do tasks that we can't or don't want to do from inside the browser for performance or security reasons. We connect to databases for example to fetch and store data. We do user authentication which we obviously can only do on a place the user can't access to make it more secure and avoid it being hacked. We do it for input validation to see if a user entered a correct e-mail address, the browser can always be tricked, users can even edit their browser side code. You can open the developer tools and start working on that page you're on but the server is of course sheltered from that, the user can't access it. And in general, we have our business logic on the server. Everything our user shouldn't see which takes too much time to run in the browser, where we obviously want to deliver a fast user experience or anything of that kind and that is where we use nodejs. So also javascript code but this time, not on the browser but on the server and this is where we use the many features nodejs gives us and this is how we indirectly allow our users to work with the server through that request response pattern, the direct access is not available. So this is how we will use nodejs in this course also, we will use it to write code on the server that returns data our users, our clients can work with. Now one important side note at this point of time, nodejs is not limited to running code on a server, it's a javascript runtime and you even saw a first demo which did not do anything where we needed a browser right, we didn't spin up a server there, we didn't do anything which we would have reached through a browser. We'll do that a lot throughout the course but we haven't done it thus far because it's just a javascript runtime, we can execute any javascript code with nodejs and often that is code that runs on a server and is executed upon incoming requests but you also often use nodejs for other code, for example for local utility scripts or build tools. If you worked with let's say react or angular or vue or anything of that kind, you actually used nodejs indirectly a lot for all the build processes these languages or frameworks needed because nodejs is a great tool for writing utility scripts. You have access to the file system so you can write and read and manipulate files and this allows you to do a lot of utility stuff on your computer that is never exposed to the public and I just want you to know that and I'll even have a section this course where I dive a little bit more in such build tools and non server side language usages of nodejs. In general and that is the most popular thing you do with nodejs though, you use it in the context of web development and server side code. So you use it to run a server and actually and that is an important difference to PHP for example, with nodejs you don't just write the code that is running on your server, you also write the server yourself, so the code that takes the incoming requests and routes them to your well other code. In PHP, you have extra tools like apache or nginx which run the servers which listen to incoming requests and then execute your php code, here nodejs does both. It does that listening and it then also does whatever you want to do in your code, so that's important and that's something you'll see in action soon. We also use it or we therefore also use it to run all our business logic, so not just to listen to incoming requests but to then work with the requests data, work with files, work with databases, all that fun stuff nodejs is capable of and we'll do all that in this course obviously. And we also handle the response side not just incoming requests, you will also learn how you use nodejs to send back data to your clients, be that html pages, html pages with dynamic content or data only in the format of json or xml or even files. So this is what we use nodejs and what we will dive heavily into in this course. Alternatives to nodejs would be things like Python, also with frameworks like flask or Django or PHP with frameworks like laravel maybe or standalone vanilla PHP of course and more, asp.net, Ruby on Rails, all that stuff, these basically are all replacements for nodejs or nodejs can be a replacement for them and there is no clear winner. All these languages are capable of doing the same kind of stuff and of course they differ in some technical regards but in general, it's great to have that broad variety. The huge advantage or one huge advantage of nodejs is that it uses Javascript, a language which you need so much in modern web development for all the frontend, for some build tools and if you then can also use it on the server side, you don't need to learn a bunch of different languages, you can use one and the same language and then use that for your server side code too. This is why nodejs is a great language to learn, you get so much efficiency out of it, it's also a highly performant and popular language. There are so many jobs out there for nodejs, there never was a better time to learn it. It's used in so many environments, also for a lot of cutting edge stuff but in general, nodejs is a great solution, it's trending, it's fast, it's efficient and it makes sure that you only need to learn one language to write all the code you need in a modern web application.

