Azat: Hello everybody. Welcome to Node Program. My name is Azat. We'll start with some introductions. We're here at Hack Reactor, which is a Coding Academy. I used to be an instructor here. I'm writing a blog, webAppLog.com. Some of you might have heard about it, if you're interested in Node.js. I've have also worked at startups, federal agencies like National Institutes of Health, the Federal Deposit Insurance Corporation, works with startups and just big companies and corporations. For the past three years my main focus was Node.js and that's where I spend most of my time and energy. It's really cool to see Node.js coming and becoming more mainstream, and more and more companies, not only startups, but large companies also adopting Node.js as their main platform and language for web applications.

So this training started about two years ago. It had different names, different reincarnations, and I taught it at also big companies like Cisco, and I was a consultant for [inaudible 00:01:26] University teaching similar kind of journey, but this is a public format, a bit re-branded format. That's my short story. You can find me on Twitter @azat_co.

This is our agenda for the first day. We'll start with short introductions. We'll just go in circle. Everybody will tell what they're doing, what they did, what they plan to accomplish this weekend. Then we'll cover some JavaScript fundamentals. After teaching this training for a few times, I noticed that some people have very different backgrounds, for example, job engineers, they come and they assume that they don't have to learn JavaScript, because it looks very easy, which is not true. There are some tricky parts in fundamentals of JavaScript that might cause big problems later on, and big misunderstandings. So we'll overview those differences, we'll take a short break, then cover Node.js fundamentals and where it is different from JavaScript, from the browser JavaScript.

I'll give an overview of some debugging tools, and some dev tools, so you can start using them if you don't use them already. We'll talk about NPM and some core modules as well. Then we'll start doing a few workshops from the Node school, so it's a hands-on training. I will talk for a little bit and then you will work on some practical aspects in the workshops. We have David here. He is a TA, teaching assistant. He will help to get on stock if you have some issues, like [inaudible 00:03:24]. After the lunch... the lunch will be approximately from 12:00 to 1:00. After the lunch we'll start writing our first Node.js applications, starting with the Hello, World Server, and try to deploy [inaudible 00:03:40] immediately. We have a few more workshops for more advanced students who accomplish more things faster. I'll be here to help.

Second day will be devoted to MongoDB and Express.js, and other advanced objects that you might have. And, again, everybody has a little bit different level. I try my best to gather people that already have development and programming background, but if you feel like this is too easy for you, I can always give you more advanced examples. Or if you have your own projects already, you can just work on them and use me as a coach. So I'll be like consulting, helping and

guiding you when you will work on your custom projects. So second day, by default, it's MongoDB and Express.js. Those are the three main components in the MEAN stack. We'll talk a little bit about Jade and Stylus, which go hand in hand with Express.js. As the results you should get at least REST API server. The last part, the first part after the lunch, last part of the day, will be devoted to, I call them, elective parts.

There are three elective parts. One that build in full stack application, so we'll build in the front end application to talk with your REST API server. Another is deploying on infrastructure of the servers, something like AWS. So it's a deployment integration and full stack integration and building sign in with Twitter or Facebook, use them all to integrate with the third party service provider. You can choose either one of them. If you're very ambitious, you can choose all three of them and try to finish them in four hours. Then we'll conclude our two-day training, but you have one week, till next Saturday, to continue working on these projects and maybe some other projects. Then we will meet again on Saturday. If you cannot attend, you can join online. Basically, we'll have like a two-hour session, Q&A session. The purpose of that, you'll have some time to process this information from the weekend and also work a little bit on your own, and then come up with questions. And maybe you can stack and then we'll try to debug your problem. Makes sense?

I get very easily distracted, so I ask you, if you have some questions, we'll wait 'til the end of the section. So it will be all like from top to bottom, and then go left to right, so wait till we go to the right, to the next topic. Because, I may already cover it in the future slide. And if you still have that questions, ask me then. So any questions right now? [laughter] We're moving. So, before the fundamentals, let's go and introduce myself. I'm Azat. I'm a software engineer. Now, what I'm trying to accomplish with this training is to teach you guys Node.js fundamentals and basics, and coach you, provide you some guidance.

So, what is JavaScript? JavaScript is the most popular language in the world, most popular programming language in the world, because its ran in every browser. It's started back in 1995 at Netscape, they call it LiveScript, but then Java was all the hype. They decided to rename it into JavaScript. So actually it has very little to do with the Java itself. Why JavaScript become and is so popular now, and then became popular. The main intention for those people were that regular people, not experienced programmers, so they can develop on browsers. So they made the language very expressive. Expressiveness in programming languages means that there is very little overhead when you write code. So, for example, in Java when you write code, you have to write a lot of keywords, like class interface, then extend that interface. You have to do a lot of preparation, even before you begin the actual writing, the actual logic for the code itself. Code for the logic.

In JavaScript it's very close to natural language, to English language, and it's very easy for people to start writing them. And actually, I think that's one of the

reasons that JavaScript is so misunderstood, because experienced programmers that worked with Java or PHP, they look at the JavaScript code, and they look, "Oh, I understand this. I can change it. I can modify it. I can make it work," without first learning the fundamentals." And then they go into pitfalls, like a global scope linkage or type in. So it's good to know about those things. But expressiveness is absolutely amazing thing. Another thing that is different in JavaScript compared to languages like Java or C++ is it slows typing. What this means is basically, if we have a string, JavaScript will try to convert it into a number, if it seesâ€¦ or vice versa. To make things complicated there are two different categories of data types. One is called primitives and one is called object. We can have a number with the lower "n" which would be a primitive, and we can have a number with the upper case "n" which will be a class. They will be treated differently by JavaScript, because this is object and this is number. If we try to compare the data types in other aspects, then most of the other aspects would be the same.

So we have four primitives; the string, number string, Boolean and regex, regular expression. And there is [inaudible 10:58] as an object. So we have string object, Boolean object, regular expression. Then we have just the plain object. Then we have functions. Functions are also objects. Another good thing about JavaScript is that it's very easy to create an object. All we have to do is to use object literal notation. Let's say; a:1, b:2, et cetera. So we use a curly brace and then we use property name, colon, and some value. The value could be... in this case the values are numbers, but it could be a string or another object, so we could nest as many levels as we want. We don't have to write anything extra. This notation became so likable and popular, so it has its own name now, it's JSON, which stands for JavaScript Object Notation, and it became de facto standard for the Rest APIs, because it replaced XML. XML had more metadata for the structure, for the property names. So this format is better, especially for mobile things when the size matters.

Function. To define a function, we can use named functions. Let's say "f". Again, we use the curly brace, but with a object literal notation. So this function definition is called named function, because we used the function's name. I usually prefer to use this approach, where I have a variable and then I assign anonymous function to that variable. So when we're going to use a name, it's called anonymous function or closure. So I assigned this value to this variable. The difference is that this would not be hoisted, so it should always precede the function and location. But if we use named function... and we can use both. So with the named function this would be valid, so that actual definition could be after the location, because this function will be hoisted. But without it, this will fail. Makes sense?

Student 1: Upon hoisting, I always had difficulty getting my head around, why does it matter? I like declaring the variables, in what I do I like to declare variables, but why does it matter that we can run the function before the function is actually spelled out? Why does that matter? How does that cause problems in

code hoisted?

Azat: I think about hoisting as a feature. So it's like an addition to make someone's life easier. Personal, I prefer just not to use it. So I would just always use this approach, because then I know where to look for the function. I should look before the location, probably in the beginning of the [inaudible 00:15:05].

Student 2: So it's not a bug, it's a feature.

Azat: I think, yeah, but this is a question for Douglas Crockford and other designers of JavaScript, why they designed it. Probably there's a good explanation for that. I didn't research about that, but if you want, you can [inaudible 00:15:20].

Functions are also first class citizens in JavaScript. So what that means, we can treat them as any variables or other objects. So we can pass with the "f" of the function, and then we can pass "f2" to it. So we can pass functions as a parameter to other functions. We will use a lot of this patterns for callbacks, to implement the synchronous code in Node.js. If you worked with jQuery, you already implemented this, because when you create a click event for a button, and then you specify what that event should do when the button is clicked, that's exactly what you did, you just passed some function to another function.

Arrays. When objects are very easy to define, we use square braces. The difference between an array and an object is that they're very similar in the way they're implemented inside of the JavaScript engine. In fact, if we try to access array, let's say this is... let's start with zero, always zero based index. Let's say we want this dozen, right? So we'll square braces and then we put the number, the index, here. If we have an object, we would access it very similarly with square braces, and inside those square braces we provide the property name. So it just happens that arrays use numbers as properties. So think about it as an object that instead of "a" and "b", instead of strings, it has numbers. And then the way we access them is very similar.

Another property of arrays that is different from object is that this value is assorted, so the order matters. Right here order doesn't matter, but here order matters. So this is helpful if you're creating a list of items that needs to be sorted in a particular order. Let's say you have comments or posts that must be in a certain order. And then we have a special class, array with capital "A," which all arrays extent from this class. This class contains a lot of useful functions for arrays. For example, slice, length, et cetera, et cetera, map. You don't have to remember all of them, but if you know about them, you can just quickly Google and then apply it to your code, so your code become less/more compact and easily readable. It's very good to know these functions.