<https://www.infoq.com/interviews/node-ryan-dahl/>

[翻译](https://blog.csdn.net/zhangxin09/article/details/6196746?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-6196746-blog-83512735.pc_relevant_aa&spm=1001.2101.3001.4242.1&utm_relevant_index=3)

[InfoQ Homepage](https://www.infoq.com/?itm_source=infoq&itm_medium=breadcrumbs_feature&itm_campaign=breadcrumbs" \o "InfoQ Homepage) [Interviews](https://www.infoq.com/interviews?itm_source=infoq&itm_medium=breadcrumbs_feature&itm_campaign=breadcrumbs" \o "Interviews) Deep Inside Node.Js With Ryan Dahl

[Microsoft JDConf ’22: Cloud Native Java, JVM Configuration, GraalVM, Spring Boot, and More - Watch All Sessions On-Demand](https://www.infoq.com/url/t/c47d51dd-4f13-4c43-8328-5eb358cea416/?label=Microsoft-EventPromoBox" \t "https://www.infoq.com/interviews/node-ryan-dahl/_blank)

# **Deep inside Node.js with Ryan Dahl**

Ryan Dahl is a programmer at Joyent and creator of Node.js. Even before Node, his focus in the past few years has been on web servers. Ryan grew up in Rochester, New York but now lives in San Francisco.

### About the conference

QCon is a conference that is organized by the community, for the community.The result is a high quality conference experience where a tremendous amount of attention and investment has gone into having the best content on the most important topics presented by the leaders in our community.QCon is designed with the technical depth and enterprise focus of interest to technical team leads, architects, and project managers.

Transcript**[InteractiveShow AllHide allFull Page Transcript](javascript:;)**

1.[Hi Ryan, would you like to introduce yourself?](javascript:;" \o ")

Yes. Thank you. I am Ryan Dahl, I work at a company called Joyent which does hosting services and I made Node.js which I assume is why I am being interviewed.

2.[What was your motivation for beginning to work on Node.js? What problem were you trying to solve and what was the solution you came up with?](javascript:;" \o ")

I was working a lot with Ruby web servers. Originally I am from a math background so I only got into computing a little while ago and I got really interested in Ruby and was fascinated by this problem of uploading files and giving a progress bar to the person uploading the file and it just amazed me that this was such a hard problem because obviously the web browser actually has like the data to display that stuff, but it's just not accessible from the DOM so just what you do is send a request to the web server and ask it how much of this file have you received and it comes back and then you update the DOM with this thing. And there is a nice little module for Mongrel which did this.  
  
I mean traditionally you need to write some sort of module for Apache or something like this and I was just impressed that Mongrel was kind of this functioning web server in a dynamic language and it impressed me how easy it was to kind of get inside of that web server. And so I would say that that is probably how I got into this and fast forward a couple of years and I think combining the kind of Javascript "arms race" that was occurring plus was I've learned about evented programming kind of seamed very natural and it was kind of an experiment but it seemed to turn out well.

3.[There are a few other projects that use server-side JavaScript like RingoJS, AppEngineJS, etc. and most of them run ontop of the JVM. What is the key difference between Node.js and those projects?](javascript:;" \o ")

Node runs on V8, obviously, which is not the JVM, more generally though these other projects kind of take traditional approach to server-side Javascript which is more or less what you see in Ruby and Python where, I always take the web server example because it is kind of the prototypical example, but you are attempting to server requests and if you have multiple clients you would actually start multiple threads to handle those at the same time. There has been this mailing list called CommonJS where they have been trying to specify how Javascript should look on the server-side and there is actually quite a few projects like Narwal, Ringo, which was formerly called something else, Helma and G which was also the next generation of that.  
  
So there has been a lot of server-side Java script things. The thing that is actually different, I mean of course it's on a new VM and stuff is that Node kind of embraces that fact that Javascript is actually inherently a single ,threadless sort of environment and kind of this purely non-blocking approach rather than kind of this traditional thing so that is the biggest difference.

4.[Since most server-side developers are not used to asynchronous APIs, would you like to describe how they work for Node.js? How are things implemented regarding disk I/O, interaction with a DB or a RESTful services?](javascript:;" \o ")

Everything is a callback. So where you would traditionally say: "Access the database, write to file, move file over there and do something else" you kind of do these sequential sort of actions one after another. In Node you can't do those sort of things because you might take some amount of time for you to move a file from one place to another because the disk might have to spin, or if you query your database that might take some milliseconds for you to respond and in Node everything is non-blocking and so it doesn't allow you to just sit there and then return the response.  
  
You have to supply a callback and so there are many anonymous functions in Node where you are giving a callback to get a response, which is disconcerting to people who are used to this traditional sort of blocking threaded server. However, I think it's a style sort of thing that you can get used to.

5.[Also it's probably very familiar to front-end developers.](javascript:;" \o ")

Right. I think that is the main selling point of Node is that we've actually been like training a generation of programmers to do exactly this and they know that when you are making XHR that you have to do it asynchronously. Everybody knows that you don't do an XHR synchronously because it locks up the webpage. But somehow on the server, of course I am going to do an XHR synchronously; I am going to request something from a database. It has to be synchronous and if it's not I don't even know how I could program that.

6.[From watching a few of your presentations I got the impression that you were probably experimenting with other languages before Javascript, but somehow the syntax or V8 maybe seemed more tempting?](javascript:;" \o ")

I was into Ruby for a while, I said, but eventually the VM just drove me insane, because it's just so slow, every time you try to make it faster you realize: "OK, I am just going to write this part in C" and every single line of Ruby that you add to your application, actually just slows down the server noticeably. And so eventually I just kind of ended up with this big C thing that I was kind of happy with, like OK I could write a web server with it and I could do file IO and so I had for a long time the idea of "I would have this kind of library that abstracted some part of the problem away and people can write it in C and they can write their little web servers.  
  
People don't like writing stuff in C. Basically I want to put people in this non-blocking environment because it is the right way to design a server, it just kind of maneuvering around all the other pieces of the computer system and I would really like to do such a thing in Haskell or some sort of declarative language like this, where you could literally be purely functional when you receive events off of the socket, because all your side effects would happen on the event loop and you would just allow that to happen and then you kind of get a function call with some data and you would do whatever with that, make some call which would not have a side effect, you just write something to some buffer that would get flushed to the kernel and then you drop back down to the event loop.  
  
Side effects would happen, everything would happen and then you get another call from that. But when you are receiving an event from the event loop you could be purely functional, you could really have nothing to do with anybody else and that is attractive. But you look in the GHC code and it's very hard and I am not such a good programmer and I gave up on that. And then V8 came out and it just kind of clicked. I am not a Javascript originally, have nothing to do with that really, but it just seams like a natural sort of thing once I started poking around VA.

7.[I was actually expecting for you to say that you would rather do it in Ruby and Haskell was a bigger surprise for me. Also the selling point for Node.js is usually that it's Javascript and you now just clarified that it's not actually what is so important about, it's just an aspect of.](javascript:;" \o ")

It's unclear if that is an important aspect, for me it's not an important aspect. It's nice that Google is producing this VM and it's BSD licensed and it's super good and there is a lot of work going into those things and that is important for. And what is really nice for that Javascript is that it had no preconceived notions of I/O. Of course there were some things and we talked about that, but generally there is nothing attached to Javascript. If you talk about Ruby, there is a puts function already, so you are kind of screwed.  
  
And another language I was looking at a lot was Lua, but it also kind of already had a standard library and was really attractive about Javascript was that it was just this pure language, just like adding numbers and strings and some anonymous functions and it's just very simple and pure and that was very attractive. But I would guess that the reason that Node received so much attention is that it is in Javascript and people have these ideas about not having their developers context switch between Javascript and JVM language and blah and doing all these sort of things.  
  
And so I think that is an attractive selling point to the people, but for me it's just kind of small language to use.

8.[For what kinds of applications do you see people using Node.js? What are the use cases that make Node.js shine?](javascript:;" \o ")

The problem that it solves right now that actually doesn't have a good solution are like little web socket servers. Things like a little game, or you have a bunch of people walking around in a room and you kind of have to relay the event that you are walking and kind of sent it out to all the other people and so people are using it a lot for this, or say a chat room or something like that because there is actually no real good ways to do this right now.  
  
I mean of course you can use various technologies to do this, but it seems that Node is kind of a nice simple solution for those kinds of things. That said, I mean, I think Node can be used in a lot of different ways, even for a traditional sort of request-response, response-database query-response sort of websites, but also maybe for like sensor networks or something like that.

9.[Like Realtime.](javascript:;" \o ")

10.[There was a recent post on Yahoo Email blog that mentioned that Node.js was considered for Yahoo mail. Do you know anything more about that? Are there any other big deployments out there?](javascript:;" \o ")

Yahoo likes Node a lot and they are actually interested in using it for several projects I think and kind of experimenting with it, the YUI group is also pretty heavily because Yahoo is really into this idea of "degregation", simple degregation, progressive degregation or something like that, where if you go to Yahoo, without Java script, it still displays with less features.

11.[Does is still work in 2010 if I go to [their site]?](javascript:;" \o ")

Supposedly, that is their theory, I've never tried it. So what they want to do is they want actually want to take their YUI, their front end Java script library and render HTML on the server if necessary. If somebody connects to it and they realize this is an old web browser, this is a web browser with Javascript disabled, we can render it on the server-side, send the Java script, send actual HTML to the server. So that is one of their projects. I think they are just generally look at it for a possible platform, for building things as well.

12.[Node.js uses event-based programming. Would you like to explain to us what that means and what are the challenges that come with that?](javascript:;" \o ")

Yes. I mean it's different. I think one of the big challenges is that you are always loosing your state. If you have threads you kind of build up this call stack and then you switch to some other and do something else and if you hit some exception somewhere, usually have this history of how you got to this thing and more generally in this kind of single threaded environments you are destroying your state, you are always going back down to the event loop and then something else happens.  
  
And so you run into these situations where you hit an exception and you get the line number and you find out where you were from, but you don't know necessarily how you arrived at that situation because the history of that is gone. So I think that is a real problem in event based programming. I think a lot of people have a hard time changing style from the synchronous point of view to the callback sort of view point and they imagine that this is an intractable problem, but it's not, they just need to get used to it.  
  
So I think there are some real problems and there are some imagine problems with this. It's not the best environment for doing large batch operations where you are one thing after another because you kind of do this sort of thing. So there are ways to get around that, of course, but it's better for this sort of situations where something comes in here, goes over there, takes up this one and events kind of occurring spontaneously that you have to handle rather than doing a shell script sort of deal.

13.[What would be a complete developer's tools stack for Node.js. Starting from an editor, debugging facilities and tools for testing, deployment and possibly monitoring?](javascript:;" \o ")

Node is by design a very small executable that actually doesn't like spew files across the file system; it's all kind of packaged up in one file. So you have to access it from the command line, but basically you can use any editor that you want to.

14.[What are you using?](javascript:;" \o ")

I use Vim. People use various things. V8, as I said, is a really great VM and they are doing a lot of work. All the debugging stuff is there in V8 and everything that you can do through kind of the Chrome developer tools which is like stepping to code and profiling your code and doing all that stuff is all available in Node as well. You need a client to kind of talk to that and so there are various projects trying to develop Chrome tools basically, but outside of that. So there is one called node-inspector that is really nice; there is an Eclipse plug-in that does that, which is actually supposed to be used for web site stuff as well.  
  
There is one called NDB (No Debugger). So we've got Editor and Debugger and what's more ...Monitoring deployment, this sort of stuff is not there and what people use is Monit and they kind of have their own custom solutions.

15.[I've seen in your chat app you basically print out the available memory.](javascript:;" \o ")

Yes. My chat application is just running in a screen thing actually. It's actually kind of stable, which is surprising. SSH did it in one day and started a little screen thing. So I put zero effort into that at all, but people who are actually concerned about their website being up, because obviously they crash, would run Monit or something and at Joyen we use Solaris, we use SMF, so I think anything that you would use for a traditional web sort of thing.

16.[Would you like to explain to us about the various ways a developers can control flow in a Node.js app, like callbacks, event emitters and promises?](javascript:;" \o ")

So I said like doing this kind of serial actions is kind of difficult because you create file-intend->callback, write the file->callback. You kind of tend to indent very far if you are doing a bunch of serial actions. So there are ways around this and so for example Tim Caswell has a library called Step which basically queues these things up. So you kind of say the functions and the arguments and more or less put them in a ray and it queues them up and kind of shuffle them out as necessary and you can kind of insert callbacks where necessary if you want to or just add the end of this action.  
  
So it's kind of interesting. I mean correct me if I am wrong, I think Node is kind of unique in that, it's the single thread environment, but it also saying we are going to be a single execution stack environment, we are actually not going to have green threads, we are not going to have code routines, you will destroy your stack every time you go to the event loop. So the people are required to kind of come up with new abstractions to deal with this problem, how do you deal with this bunch of problems that need to be executed in sequence. So it's kind of an interesting problem, but there is a couple of them.

17.[Would you like to explain to us what is the current way to do load balancing for a Node.js application? What are the plans for the future?](javascript:;" \o ")

Yes. Node provides this low level sort of networking infrastructure. It doesn't attempt to solve problems of scaling out over multiple CPUs or multiple data centers or multiple machines. This is left up to the user to kind of decide for themselves and I mean this is a hard problem. If you have a chat server and you are running it in one process and you realize that you are running up to 20000 people and now garbage collection is becoming compute bound and the server is becoming slow, you say: "Ok, I am spending all my time in GC, it would be nice if I could use this other core that's like sitting idle right now."  
  
So now you can start another process, another Node process, which generally the answer is start more processes, let the kernel schedule these things to different. But I mean you have to talk about the specifics of the problem. So if it's a chat application you have to start another core and now you can have people connect there and people connect there, but you also have to connections in between, like an IRC server network, you are going to have to kind of have a fat pipe in between them and send a bunch of data between.  
  
If you just have, say a simple website where you're just request-database-response sort of thing and all the connections are independent of each other and they don't have to talk to each other, then scaling out is very easy. You just start a bunch of them and load balance across it in any way that you want to. You can stick an Nginx server in front of it, you can IP, DNS load balance or whatever, you can do this trick to actually start one server socket and then "fork" multiple do it in quotes. Because it's not actually forking, but you can make a prefork server where you fork your process times and being the number of cores that you have.  
  
And then you kind of get a copy of this file descriptor on each one of them and so all of these cores are kind of looping trying to accept connections on this socket. And when an incoming connection comes whichever one of those guys has scheduled they are kind of racing to do this and so load balance seems kind of done kernel like Nginx workers work that way. So generally it just depends on the problem. I mean scaling out something can either be easy, depending if it's an easy problem or it can be very hard.

18.[There are many micro benchmarks out there that compare Node.js performance with Nginx, JavaScript on top of the JVM, etc. There are even more people that give interpretations of the results from these benchmarks and draw all kinds of conclusions, like how better the garbage collection is on the JVM than on V8, that Node.js performance degrades with large packet size's, etc. What is your opinion about Node.js performance?](javascript:;" \o ")

Generally these benchmarks on the internet are not so interesting. There is a couple of them that are actually interesting. Here is the performance situation: Node is pretty fast and it's kind of at the same order of magnitude as any other event loop system like this. I mean you kind of get this big jump when you go from a system like Ruby on Rails to Event machine.

19.[With Ruby on Rails you get a big jump doing anything.](javascript:;" \o ")

Yes, doing anything. But I mean like going from the traditional sort of web framework to event machine twisted you get a big jump. You also get a big jump even in C where you go from a server that handles connections with OS threads to a system that handles connections in an event loop. So you kind of get this big jump, but when you compare these different systems, more or less it's the same. Node basically serves as fast as Event machine and in terms of like a "hello, world" server, sort of situation. So it's in that category of like: Ok, now it's evented and it's kind of fast.  
  
It can handle many connections fairly well, you can load it up with Idle TCP to whatever limit your OS provides and it will sit there idle, which is a good sign, but not so interesting. You can load up several thousand connections, let's say 20000, and have them all writing a small amount of data to the server and getting it echoed back but not to each other and it will handle that sort of situation with 10s of thousands of clients. If you get in more complicated situations, where, you know, one guy is talking to all of the clients the performance degregates very quickly because now you have to write to n sockets.  
  
Anyway that would be the same situation in any system. Anyway performance is fairly good. What is not good is pulling strings out of V8. So writing a string to a socket, small strings are OK, but larger strings, dumping them to the socket is very slow. So say a 10 kilobyte string, if you want to write it to a socket it actually performs very poorly, which is surprising, like nobody has done this test on benchmarks and said: "Look at that, that really sucks", because it really does suck. I think there are some ways around this problem.

20.[Is it the fault of V8 or what?](javascript:;" \o ")

The problem is that their systems can go into the heap of VM or whatever they are doing and write that directly to the socket and V8 forces me to copy it out into a separate buffer and then send it out. So that extra copy kills you on large strings. But there is a way to solve that, which is going to the VM and actually copy it out from that and so these things can be solved, I think. We do see some pretty bad situations with V8. Generally I am very happy with V8, but we do see situations where garbage collection takes a second.  
  
I am very ashamed to say it, but there are some bad situations with V8 GC. Maybe it will be improved, but generally I think those are the two major things. I guess the other big concern is the 64 bit V8 is bounded by 1.7 Gigabyte heap, so that is a fairly constraining problem. I don't know if the 32 bit is bounded by anything. So you are on 64 bit, your heap is maxed out at 1.7 Gigs, so that is not so good. And the situation there is less clear to me because I don't actually know what the problem it.  
  
I guess it had something to do with when they do the compact thing they have some sort of offset for where it is and there is only a certain number of bits available for determining where those things are going to be compacted to. So that is a hard problem that is something that V8 is going to have to solve.

21.[It is generally accepted that the Javascript VMs we had up until recently maybe, up until V8 where really poor pieces of software and now V8 is only on its first generation so you suppose to see great gains in the following years?](javascript:;" \o ")

I don't know about great gains, but it's certainly going to be improved. I mean they are doing work all the time. It looks bright for its future. I mean there is a patch being discussed on the V8 mailing list that would fork out threads for doing the mark phase of the garbage collections. So you could actually use multiple course like walk the object graph. So on a server system where you might be sitting on a 32 core box that might actually help significantly. There is a lot of ideas of how this stuff can be improved and the V8 people are very smart.

22.[There is a debate amongst JavaScript developers regarding the usage of the many projects like Traits.js and JS.Class that provide their own way of doing OOP with JavaScript. These libraries are not only criticized about their performance, also on the grounds that they provide a suboptimal way of using the language's native facilities like prototypes. What is your take on this subject?](javascript:;" \o ")

I try to use Javascript as it's intended which is, you know, stick methods on the prototype thing and I am also not in favor of these libraries.

23.[Where can I host a Node.js app?](javascript:;" \o ")

You might have some problems with Free BSD, but generally you can compile it on Linux. And so any VPS things you should be able to up and running without too much problems. I work for a hosting company, Joyent, and we are working on a service called NO.DE which is a Node hosting service and it's going to be one of these "get deployment" things where you write all your code and you manage it with git and then you do git push Joyent and then it kind of manages the restarting of the server and that sort of stuff.  
  
And so that's in Beta right now and should be released in the near future, but at the moment what you are forced to do is set up your own VPS and do it, but it seems like in the near future there will be some options to kind of simplify this and kind of having "AppEnginy" sort of feel.

24.[You also recently announced the roadmap for Q4/10. Would you like to give us an overview of that and maybe give us a hint on what's to come next?](javascript:;" \o ")

Basically we are fixing things. Like I talked about this string problem, this kind of requires a bit of a refactor and so we are addressing these performance concerns and kind of generally fixing bugs and making it more stable. And I think that is mostly what my roadmap is about. After that I think we'll be fixing bugs for a long time, but there is not too much I want to add to this thing. It should be small. I mean it's the sort of thing you are not going to use by itself.  
  
You are going to have to add libraries and you are going to have to build stuff on top of it because it's a very kind of foundational sort of thing, but I want that to exist outside of the core project and so I don't see the core project changing enormously. I don't think we are going to be adding any major components to it or changing the API drastically in the future. There are just kind of touch-ups and bug fixing into the foreseeable future is about where it's going.

25.[Would you like to give us an example of a few frameworks and APIs that you think our viewers should probably check out if they are planning on using Node.js?](javascript:;" \o ")

Socket.IO seems to be extremely popular and it's a great idea which is use websockets on every browser. You can use websockets on IE and the way that it works is that modern browsers have real websockets and so you load up this Javascript file and it detects your web browser and if you have a normal web browser then you just the normal websockets. If you have an older web browser then there are various tricks you can use to get a websocket like interface, like long-polling So what it will do it will give you a websocket API which is in this browser library and do long polls to the server.  
  
And then on the server side it gives you this also websocket sort of field because it's Javascript and you are one the server now and it has various servers open. So it's HTTP server to do long polling, it's got a websocket server to handle websockets, it's got another server to do flash sockets. So it's got all these various depending on what sort of browser you are on, you are going to connect to the server in different ways. But the user of these things will only have one API to deal with. So this basically solves the problem like, OK now you don't need to know what long polling is anymore, which is great, excellent.

26.[In 10 different browsers!](javascript:;" \o ")

Yes in 10 different browsers. So this thing, I think it works on everything, but what I define everything to be IE 6 and later. So that is great and a lot of people are using that. There is a library called Connect which is somewhat Rack for Node, with the fact that Node is very asynchronous and kind of does these streaming sort of things and so you have your Node web server and then you can kind of put these blocks, these filters that get passed through your application, like a GZIP sort of filter. So it kind of gives you this nice API to put these blocks or you want a static file server to handle a certain path and you just kind of stick that filter on there and it handles it.  
  
And on top of Connect is a library called Express which is kind of Sinatra where you do app.get/ and give a callback and then and so a lot of people are using that one. You should also check out Peter Griess has a webworker library which is basically a way to create new Node processes. So you specify a Javascript file and you say start a new worker and it starts a new process that starts executing that Javascript file that you gave it and internally it has a pipe between these things that opens up and it has a nice real serialization format and so now what you can now do is start talking to that web worker.

27.[It's a completely different process?](javascript:;" \o ")

It's a real OS process. It starts a new process and so now you can start sending JSON between these things and you can say: "I need to convolve this image. Please do this gausian blur for me." And you tell it to that other worker via JSON and it does that, it's spinning the CPU super heavily; you go back you start serving your request. And so this is kind of a nice way to distribute tasks over multiple CPUs.  
  
And there is a really nice pcap library for listening to your ethernet interfaces on your computer, so you can actually get callbacks, it's a very asynchronous sort of thing already and so you can kind of get callbacks into Java script saying: "Hey, a new packet arrived on your interface. Oh a new packet is going on your interface. And so Matt Ranney who did this binding to pcap also hooked it up to the HTTP parser and it does all the TCP reordering of the packets and then sends it to the parser and now you can get events of every HTTP request that is occurring on the server.  
  
So I think those are fun libraries that you should get you started. So I should say that if you go to the Github page there is a wiki and on the wiki page there is a module's page which has many modules and you can look through there.

28.[What is your take on the whole CommonJS effort and what standards are now being supported by Node.js like Modules and how do you see this whole thing evolving?](javascript:;" \o ")

CommonJS which is like this mailing list of people who are interested in Javascript and want to kind of define APIs for it outside of the web browser defined a Module specification which is this "require" that Node uses. They are very active and they write out these huge specifications. For a long time all of their specifications were synchronous and they were just kind of thinking: "Ok, there is Ruby, there is Python let's just take the best of both worlds, put it in the Javascripts and now we have this cool language."  
  
That obviously did not go well with me because I had this idea of doing this whole non blocking thing and so none of these APIs or most of these APIs that they were interested in, including the module thing were acceptable because they were blocking, they blocked a lot and in a language like Javascript where you actually have no concept of threads, of course Rhino has threads and SpiderMonkey has threads, but these are kind of additions to the language. The language itself like ECMAScript 5 does not have threads. So we've had a lot of conversations about this.  
  
My opinion is that standards bodies are great for web browsers where there are 5 people involved and they kind of have to agree on this sort of thing, but for the server-side world where it's completely green fielded situation with zero users and we should be playing around with APIs rather than agreeing first on an API and kind of prescribing how this world should be, rather than just going and figuring it out. And so I've kind of backed away from the CommonJS group just because I've got users and we work on new APIs and we've iterated through several versions of this and we are kind of discovering what works and what doesn't and how this Javascript actually works with your operating system.  
  
And hopefully in the next year or so we will kind of converge on something that is acceptable and at that point maybe there is going to be some other server side Javascript solutions. There is already, but maybe we can get together and say: "Now we have these things, let's try to be interoperable right now." But at the moment it's a pipe trim to try to be interoperable with systems that are basically inoperable.

29.[Do you think it is realistic to hope also, I mean you talked about being interoperable between different server-side solutions, what about code that was written to be executed in the browser? Is it realistic to say: "I've got that piece of code that I can actually use on my browser" or are there actually cases like that, because most of the browser code is DOM manipulation?](javascript:;" \o ")

It's not as much as people hope. I always meet this people who are just like: "Oh, great, I can just share my web server on my browser and then everything is going well". I mean your web browser and the web server are doing very different things and usually there is not so much code that can be shared. But there are situations where you might share something like validation sort of libraries where you might want to highlight like a form field and check that they enter the email address correctly, but once they actually submit it, just so you are not believing what they what they entered on the form you can re-check before you send it to the database.  
  
And so you can imagine sort of situations where you would want to share code between the server and the client, we talked about YUI, where they are actually having the DOM on the server-side and generating code for the server and so they are sharing a lot of code there. So yes, possible, interesting, I think what will be really interesting is seeing rather than you writing code and executing it on both environments is the sort of scraping abilities, where say you are writing some sort of web scraping technology, you can pull down a website, actually run the Javascript, have your virtual DOM inside of Node, JS DOM is an implementation of the DOM for Node, have that be run, pull down the jQuery or whatever it runs and have it manipulate the DOM.  
  
Then pull out the text in that dive that you were interested in which you had no possibility of doing before, because all you could get is the HTML and these days a lot of HTML is being generated on load time by some JSON that is being loaded by an XHR. So I think for screen scraping this is really interesting. But generally, I think this sharing code between the two environments, first situations make a lot of sense, for the typical:" I am going to write a website" there is not so much code that gets shared.

30.[Thank you very much, Ryan.](javascript:;" \o ")

DEC 13, 2010

Interview with

[Ryan Dahl](https://www.infoq.com/profile/Ryan-Dahl/)

* FOLLOW

by

[Dio Synodinos](https://www.infoq.com/profile/Dio-Synodinos/)

* President at C4Media | InfoQ, QConFOLLOW