There are many great reasons to use Node.js, regardless of experience level. Take a look into what some of the greatest practical reasons are to use Node and why you should love it.

I get it. You're not a bandwagon developer. You don't use the cool, trendy platform just because everyone else is. That's why you haven't looked seriously at Node.jsyet. (Or your boss hasn't let you yet.) Well, it's time to look again. There are many great, practical reasons to use Node. Here are ten of them.

1. You Already Know JavaScript

Let me guess. You're using a rich client framework (Angular, Ember, Backbone) and a REST-ful server-side API that shuttles JSON back and forth. Even if you're not using one of those frameworks, you've written your own in jQuery. So if you're not using Node.js on the server, then you're constantly translating. You're translating two things: 1) the logic in your head from JavaScript to your server-side framework, and 2) the HTTP data from JSON to your server-side objects.
By using JavaScript throughout your app, you not only gain mental energy, you gain practicality as well. By potentially re-using your models, and templates, you reduce the size of your application which reduces complexity and chance for bugs.

JavaScript as a language is eating the world. It is not going away soon. There is a JavaScript runtime on every personal computer in the world, and it looks to stay that way for awhile.

2. It's Fast

Node.js is a JavaScript runtime that uses the V8 engine developed by Google for use in Chrome. V8 compiles and executes JavaScript at lightning speeds mainly due to the fact that V8 compiles JavaScript into native machine code.

In addition to lightning fast JavaScript execution, the real magic behind Node.js is the event loop. The event loop is a single thread that performs all I/O operations asynchronously. Traditionally, I/O operations either run synchronously (blocking) or asynchronously by spawning off parallel threads to perform the work. This old approach consumes a lot of memory and is notoriously difficult to program. In contrast, when a Node application needs to perform an I/O operation, it sends an asynchronous task to the event loop, along with a callback function, and then continues to execute the rest of its program. When the async operation completes, the event loop returns to the task to execute its callback.

In other words, reading and writing to network connections, reading/writing to the filesystem, and reading/writing to the database–all very common tasks in web apps–execute very, very fast in Node. Node allows you to build fast, scalable network applications capable of handling a huge number of simultaneous connections with high throughput.

3. Tooling

npm is the Node.js package manager and it... is... excellent. It does, of course, resemble package managers from other ecosystems, but npm is fast, robust, and consistent. It does a great job at specifying and installing project dependencies. It keeps packages isolated from other projects, avoiding version conflicts. But it also handles global installs of shell commands and platform-dependent binaries. I can't remember a time with npm where I've had to ask myself, "Why are those modules conflicting? Where is that module installed? Why is it picking up this version and not that one?"
grunt is the venerable task runner, but new kids on the block gulp, brunch, andbroccoli focus on builds that transform your files, and take advantage of JavaScript's strong file streams capabilities.

4. You Already Know JavaScript, Again

DB

So you've decided to use JavaScript on the server, and you're proud of your decision that avoids all that translating from client data to server data. But persisting that data to the database requires even more translations!

There's good news. If you're using an object database like Mongo, then you can extend JavaScript to the persistence layer as well.
Using Node.js allows you to use the same language on the client, on the server, and in the database. You can keep your data in its native JSON format from browser to disk.

5. Real-time Made Easy

If Node.js excels at many concurrent connections, then it makes sense that it excels at multi-user, real-time web applications like chat and games. Node's event loop takes care of the multi-user requirement. The real-time power comes thru use of the websocket protocol. Websockets are simply two-way communications channels between the client and server. So the server can push data to the client just as easily as the client can. Websockets run over TCP, avoiding the overhead of HTTP.

Socket.io is one of the most popular websocket libraries in use, and makes collaborative web applications dead simple. Here's a simple server using socket.io:

```
var app = require('http').createServer(handler)
var io = require('socket.io')(app);

app.listen(8080);

io.on('connection', function (socket) {

  // Send a message to the client
  socket.emit('event to client', { hello: 'world' });

  // Handle a message from the client
  socket.on('event from client, function (data) {
    console.log(data);
  });
});
```

6. Streaming data

Traditionally, web frameworks treat HTTP requests and responses as whole data objects. In fact, they're actually I/O streams, as you might get if you streamed a file from the filesystem. Since Node.js is very good at handling I/O, we can take advantage and build some cool things. For example, it's possible to transcode audio or video files while they're uploading, cutting down on the overall processing time. Node can read/write streams to websockets just as well as it can read/write streams to HTTP. For example, we can pipe stdout from a running process on the server to a browser over a websocket, and have the webpage display the output in real-time.

7. One Codebase And Your Real-time For Free

If you've made it this far, you may ask yourself, "If Node.js allows me to write JavaScript on the client and server, and makes it easy to send data between the client and server, can I write a web app that runs a single codebase on both client and server, and automatically synchronizes data between the two?"

The answer to your question would be yes, and the framework for that app would beMeteor. Meteor is a next-generation web framework built atop Node. It runs the same codebase on the both the client and server. This allows you to write client code that saves directly to a database. Then, that data is automatically persisted to the server. It works the other way too! Any data changes on the server are automatically sent to the client. It gets better. Any webpage displaying that data reacts automatically and updates itself!

```
// Save the value of 'name' upon clicking 'submit' directly in the browser!
'.click .submit': function(e, tpl) {
  Users.update(
    { _id: this._id },
    { $set: { name: $('.name').val() }}
  );
}
```

8. Corporate Caretaker

The inherent risk with any open-source project is abandonment by its volunteer maintainers. This isn't the case with Node.js. Node is currently sponsored by Joyent, who has hired a project lead and other core contributors, so there is a real company backing the future of the project. Not to mention there are a great number of major companies backing the project at every level including Walmart, Microsoft, Yahoo, Paypal, Voxer, and more.

9. Hosting

With rapid adoption, world-class Node.js hosting is also proliferating. In particular, Platform-as-a-Service (PaaS) providers such as Modulus and other reduce deployments to a single command. Even the granddaddy of PaaS, Heroku, now formally supports Node deployments.

10. Every Developer Knows (A Little) JavaScript

This one's for your boss.

Since the dawn of the web, there have been JavaScript onclick's and onmouseover's. Every web developer has coded a little JavaScript, even if that JavaScript was hacking a jQuery plugin. Finding web development talent is terribly difficult these days. So when choosing a web platform, why not choose the platform whose language is known by every web developer in the world?

In Conclusion, A Bonus!

But wait, there's more! As with any platform or product, open-source or otherwise, its community is a huge influencing factor. And Node's is second to none. From meetups to conferences, there are really smart people working on the ecosystem every day. At the same time, the community is welcoming. These same smart people are always willing to offer help to folks new to Node, or even programming in general. You won't feel bad for asking a question on IRC or opening an issue. This community is also very active, with over 91,000 modules on npm. And this community is generous. In 2013, individuals donated over $70,000 to help run the public npm servers.

Yes, Node is trendy at the moment. This is web development, so next week Node may be dead, and the next hot thing will have arrived (will it be Go or Elixir?). But give it a try.