





O que exatamente é o Node.js?



developerWorks Brasil em JavaScript, Redes e Servidores
segunda-feira, 5 de setembro de 2011

0
SHARES



Se você ouviu falar do Node, ou leu artigos que proclamam como   ele é maravilhoso, poderá estar pensando: “Afinal, o que é Node.js?”. Apesar de não ser para todos, o Node pode ser a escolha certa para algumas pessoas.

Este artigo buscará responder o que é o Node.js resumindo o problema que ele pode resolver, como ele funciona, como executar um aplicativo simples e, finalmente, quando o Node é ou não é uma boa solução. Ele não abordará como escrever um aplicativo Node complicado nem será um tutorial completo sobre Node. A leitura deste artigo o ajudará a decidir se você deverá buscar aprender Node para usar em seu próprio negócio.

Que problema o Node soluciona?

O objetivo declarado do Node é “fornecer uma maneira fácil de criar programas de rede escaláveis”. Qual é o problema com os programas de servidor atuais? Vamos fazer as contas. Em linguagens como Java™ e PHP, cada conexão inicia um novo encadeamento que, potencialmente, é acompanhado de 2 MB de memória. Em um sistema que tenha 8 GB de RAM, isto define o número máximo teórico de conexões simultâneas em cerca de 4.000 usuários.

À medida que sua base de clientes cresce, você deseja que seu aplicativo da Web suporte mais usuários e, portanto, será necessário adicionar mais servidores. É claro, isso se soma a custos de negócios, especificamente custos de servidor, custos de tráfego e custos de mão de obra. Adicione

a esses custos o problema técnico potencial de que um usuário poderá usar diferentes servidores para cada solicitação, de forma que quaisquer recursos compartilhados deverão ser compartilhados por todos os servidores. Por exemplo, no Java, variáveis estáticas e caches precisam ser compartilhados entre as JVMs em cada servidor. Este é o gargalo de toda a arquitetura de aplicativos da web, o número máximo de conexões simultâneas que um servidor pode tratar.

O Node soluciona o problema mudando a forma como uma conexão é feita no servidor. Em vez de iniciar um novo encadeamento do SO para cada conexão (e alocar a memória correspondente com ele), cada conexão cria um processo, que não requer que o bloco de memória o acompanhe. O Node alega que nunca ocorrerá um impasse de bloqueios, pois não são permitidos bloqueios e ele não bloqueia diretamente para realizar chamadas de E/S. O Node também alega que um servidor que o execute pode suportar dezenas de milhares de conexões simultâneas. De fato, o Node altera o panorama do servidor ao mudar o gargalo do sistema inteiro do número máximo de conexões para a capacidade de tráfego de um único sistema.

Portanto, agora que você tem um programa que pode tratar dezenas de milhares de conexões simultâneas, o que você pode de fato criar com o Node? Seria ótimo se você tivesse um aplicativo da Web que exigisse tantas conexões. Este é um daqueles problemas do tipo “se você tem esse problema, ele não é um problema”. Antes de chegarmos a isso, vejamos como o Node funciona e como foi projetado para ser executado.

O que o Node definitivamente não é

Sim, o Node é um programa de servidor. No entanto, ele definitivamente não é como o Apache ou o Tomcat. Esses servidores são produtos de servidor independentes, prontos para instalar e implementar aplicativos instantaneamente. Você poderá ter um servidor em execução em um minuto com esses produtos. O Node definitivamente não é isso.

O Apache pode adicionar um módulo PHP para permitir que os desenvolvedores criem páginas da Web dinâmicas, e os programadores usando Tomcat podem implementar JSPs para criar páginas da Web dinâmicas. O Node definitivamente não é isso.

Neste momento inicial da vida do Node (atualmente na versão 0.4.6), ele

não é um programa de servidor pronto para ser executado, onde você espera instalá-lo, colocar seus arquivos dentro dele e ter um servidor da Web totalmente funcional. Ele ainda requer uma quantidade de trabalho não trivial para obter até mesmo a funcionalidade básica de um servidor da Web funcionando depois de concluir a instalação.

Como o Node funciona

O Node propriamente dito executa V8 JavaScript. Espere, JavaScript no servidor? Sim, você leu corretamente. O JavaScript no lado do servidor é um conceito relativamente novo, e há cerca de dois anos, aqui no developerWorks, ele foi mencionado em uma discussão sobre o produto Aptana Jaxer. Apesar de o Jaxer nunca ter chegado a tanto, a ideia em si não era tão absurda — por que não usar no cliente a mesma linguagem de programação que você usa no servidor?

O que é o V8?

O mecanismo V8 JavaScript é o mecanismo subjacente do JavaScript que o Google usa com seu navegador Chrome. Poucas pessoas pensam sobre o que de fato ocorre com o JavaScript no cliente. Um mecanismo JavaScript, de fato, interpreta o código e o executa. Com o V8, o Google criou um interpretador ultrarrápido escrito em C++ que tem um aspecto exclusivo: é possível fazer o download do mecanismo e integrá-lo em qualquer aplicativo que você desejar. Ele não é restrito à execução em um navegador. Portanto, o Node, na verdade, usa o mecanismo V8 JavaScript escrito pelo Google e o redireciona para uso no servidor. Perfeito! Por que criar uma nova linguagem quando há uma boa solução já disponível.

Programação direcionada a eventos

Muitos programadores foram ensinados a acreditar que a programação orientada a objeto é o projeto de programação perfeito e a não usarem nada mais. O Node utiliza o que é chamado de modelo de programação direcionado a eventos.

Listagem 1. Programação direcionada a evento no lado do cliente com jQuery

// jQuery code on the client-side showing how Event-Driven programming works

```
// When a button is pressed, an Event occurs - deal with it
// directly right here in an anonymous function, where all the
// necessary variables are present and can be referenced directly
$("#myButton").click(function(){
    if ($("#myTextField").val() != $(this).val())
        alert("Field must match button text");
});
```

O lado do servidor, na verdade, não é diferente do lado do cliente.

Verdade, não é preciso pressionar botões, nem digitar em campos de texto, mas, em um nível mais alto, eventos estão ocorrendo. Uma conexão é feita – evento! Dados são recebidos pela conexão – evento! Dados param de chegar pela conexão – evento!

Por que este tipo de configuração é ideal para o Node? O JavaScript é uma excelente linguagem para programação direcionada a eventos, pois permite funções e fechamentos anônimos e, mais importante, a sintaxe é familiar para quase todos que alguma vez já programaram. As funções de callback, que são chamadas quando um evento ocorre, podem ser escritas no mesmo local onde você captura o evento. Portanto, é fácil de codificar, fácil de manter, sem estruturas orientadas a objetos complicadas, sem interfaces e sem potencial para excessos na arquitetura. Basta aguardar um evento, escrever uma função de callback e a programação direcionada a eventos toma conta de tudo!

Exemplo de aplicativo Node

Finalmente vamos ver algum código! Vamos juntar tudo o que discutimos e criar nosso primeiro aplicativo Node. Como vimos que o Node é ideal para tratar aplicativos com alto tráfego, vamos criar um aplicativo da Web muito simples, criado para oferecer velocidade máxima.

Eis as especificações de nosso aplicativo de amostra transmitidas pelo “chefe”: criar uma API ReSTful geradora de números randômicos. O aplicativo deverá receber uma entrada, um parâmetro chamado “number”. O aplicativo, a seguir, retornará um número randômico entre 0 e este parâmetro, e retornará o número gerado para o chamador. Como o “chefe” espera que esse aplicativo seja extremamente popular, ele deverá tratar 50 mil usuários simultâneos. Vamos ver o código:

Listagem 2. Gerador de número randômico do Node

```
// these modules need to be imported in order to use them.
// Node has several modules. They are like any #include
// or import statement in other languages
```

```
var http = require("http");
var url = require("url");

// The most important line in any Node file. This function
// does the actual process of creating the server. Technically,
// Node tells the underlying operating system that whenever a
// connection is made, this particular callback function should be
// executed. Since we're creating a web service with REST API,
// we want an HTTP server, which requires the http variable
// we created in the lines above.
// Finally, you can see that the callback method receives a 'request'
// and 'response' object automatically. This should be familiar
// to any PHP or Java programmer.
http.createServer(function(request, response) {

    // The response needs to handle all the headers, and the return codes
    // These types of things are handled automatically in server programs
    // like Apache and Tomcat, but Node requires everything to be done yourself
    response.writeHead(200, {"Content-Type": "text/plain"});

    // Here is some unique-looking code. This is how Node retrieves
    // parameters passed in from client requests. The url module
    // handles all these functions. The parse function
    // deconstructs the URL, and places the query key-values in the
    // query object. We can find the value for the "number" key
    // by referencing it directly - the beauty of JavaScript.
    var params = url.parse(request.url, true).query;
    var input = param.number;

    // These are the generic JavaScript methods that will create
    // our random number that gets passed back to the caller
    var numInput = new Number(input);
    var numOutput = new Number(Math.random() * numInput).toFixed(0);

    // Write the random number to response
    response.write(numOutput);

    // Node requires us to explicitly end this connection. This is because
    // Node allows you to keep a connection open and pass data back and forth,
    // though that advanced topic isn't discussed in this article.
    response.end();

    // When we create the server, we have to explicitly connect the HTTP server to
    // a port. Standard HTTP port is 80, so we'll connect it to that one.
}).listen(80);

// Output a String to the console once the server starts up, letting us know everything
// starts up correctly
console.log("Random Number Generator Running...");
Iniciando esse aplicativo
```

Coloque o código acima em um arquivo chamado "random.js". Agora, para iniciar esse aplicativo e executá-lo (portanto, criar o servidor HTTP e aguardar conexões na porta 80), simplesmente execute o comando a

seguir em seu prompt de comando: `% node random.js`. Eis o que se parecerá quando o servidor estiver em execução.

```
root@ubuntu:/home/moilanen/ws/mike# node random.js
Random Number Generator Running...
Acessando esse aplicativo
```

O aplicativo está em execução. O Node está aguardando conexões neste momento, então vamos testar o aplicativo. Como criamos uma API RESTful simples, podemos acessar o aplicativo usando nosso navegador. Digite o seguinte endereço (assegure-se de ter completado a etapa anterior): `http://localhost/?number=27`.

A janela de seu navegador mudará para um número aleatório entre 0 e 27. Pressione recarregar em seu navegador e obterá outro número randômico. E aí está, seu primeiro aplicativo Node!

Node, para que ele serve?

Depois de ler tudo sobre o Node, você poderá responder o que ele é, mas ainda imaginará quando deverá usá-lo. Essa é uma pergunta importante a fazer, pois existem certas coisas para as quais o Node é bom e, de forma contrária, há certas coisas para as quais o Node, no momento, provavelmente não é uma boa solução. Você precisa decidir cuidadosamente quando usar o Node, pois usá-lo na situação errada poderá levar a MUITA codificação extra.

Para o que ele é bom

Como você viu até agora, o Node é extremamente bem projetado para situações em que um grande volume de tráfego é esperado e a lógica e o processamento necessários do lado do servidor não são necessariamente volumosos antes de responder ao cliente. Bons exemplos de onde o Node seria excelente incluem:

- **Uma API RESTful** – Um serviço da Web que forneça uma API RESTful recebe alguns parâmetros, interpreta-os, monta uma resposta e envia-a (normalmente uma quantidade relativamente pequena de texto) de volta ao usuário. Esta é uma situação ideal para o Node, pois você poderá criá-lo para tratar dezenas de milhares de conexões. Ela também não requer um grande volume de lógica; ela simplesmente procura valores em um banco de dados e monta uma resposta. Como a resposta é uma pequena quantidade de texto e a solicitação de entrada é uma pequena quantidade de texto, o volume de tráfego não é grande, e um computador poderá provavelmente tratar as demandas de API mesmo da API da empresa mais movimentada.

- **Fila do Twitter** – Pense em uma empresa como a Twitter, que precisa receber tweets e gravá-los em um banco de dados. Existem literalmente milhares de tweets chegando a cada segundo e o banco de dados não consegue acompanhar o número de gravações necessárias durante os momentos de pico de uso. O Node torna-se uma engrenagem importante na solução deste problema. Como vimos, o Node consegue tratar dezenas de milhares de tweets que chegam. Ele pode gravá-los rápida e facilmente em um mecanismo de enfileiramento em memória (memcached, por exemplo), a partir do qual outro processo separado pode gravá-los no banco de dados. A função do Node é rapidamente coletar o tweet e passar essa informação para outro processo, responsável por gravá-lo. Imagine outro projeto — um servidor PHP normal que tenta tratar gravações no banco de dados em si — cada tweet causaria um pequeno atraso ao ser gravado pelo banco de dados, pois a chamada ao banco de dados estaria sendo bloqueada. Uma máquina com este design só poderia ser capaz de tratar 2000 tweets por segundo, devido à latência do banco de dados. Um milhão de tweets por segundo requer 500 servidores. O Node, em vez disso, trata cada conexão e não bloqueia, possibilitando que ele capture o máximo de tweets possível. Uma máquina com Node, capaz de tratar 50.000 tweets por segundo, requer somente 20 servidores.
- **Servidor de arquivos de imagem** – Uma empresa que tem um grande Web site distribuído (pense no Facebook ou Flickr) poderia decidir dedicar servidores inteiros a simplesmente servir imagens. O Node seria uma boa solução para esse problema, pois a empresa pode usá-lo para codificar um recuperador de arquivos fácil e, a seguir, tratar dezenas de milhares de conexões. O Node procuraria pelo arquivo de imagem, retornaria o próprio arquivo ou um erro 404 e não faria mais nada. Essa configuração permitiria que esses tipos de Web sites distribuídos reduzissem o número de servidores necessários para servir arquivos estáticos, como imagens, arquivos .js e arquivos .css.

Para o que ele não serve

É claro, o Node não é a escolha ideal em algumas situações. Eis alguns cenários em que o Node não seria bom:

- **Páginas criadas dinamicamente** – Atualmente, o Node não fornece uma forma padrão para criar páginas dinâmicas. Por exemplo, ao usar a tecnologia JavaServer Pages (JSP), é possível criar uma página index.jsp que contenha loops em snippets JSP, como `<% for (int i=0; i<20; i++) { } %>`. O Node não permite esses tipos de páginas dinâmicas direcionadas a HTML. Novamente, o Node não é idealmente adequado para ser um servidor de páginas da web, como o Apache e o Tomcat o são. Portanto, se quisesse fornecer uma solução no lado do servidor para isto no Node, teria que codificar a solução inteira você mesmo. Um programador PHP não gostaria de programar um conversor PHP para o Apache toda vez que implementasse um aplicativo da web, mas, neste momento, é o que o Node exigiria que você fizesse.

- **Aplicativos pesados em bancos de dados relacionais** – O Node foi projetado para ser rápido, assíncrono e sem bloqueio. Os bancos de dados não necessariamente compartilham desses objetivos. Eles são síncronos e com bloqueio, pois chamadas ao banco de dados para leitura e gravação bloqueiam até que um resultado seja gerado. Portanto, um aplicativo da Web que solicite muitas chamadas ao banco de dados, muitas leituras e muitas gravações com cada solicitação seria uma aplicação ruim para o Node, pois o banco de dados relacional em si estaria negando muitos dos pontos fortes do Node. (Os novos bancos de dados NoSQL são uma escolha melhor para o Node, mas este é um tópico totalmente diferente).

Conclusão

A pergunta “O que é Node.js?” deve estar respondida. Depois de ler esse artigo, você deverá ser capaz de explicar, em poucas frases claras e concisas, o que é o Node.js. Se puder fazer isso, você está à frente de muitos codificadores e programadores. Muitas pessoas com quem conversei sobre o Node ficaram confusas sobre o que exatamente ele faz. Eles estão, compreensivelmente, com a postura mental do Apache — um servidor é um aplicativo no qual você coloca seus arquivos HTML e tudo funciona. O Node é direcionado a finalidade. É um software que usa JavaScript para permitir que os programadores rápida e facilmente criem servidores da Web rápidos e escaláveis. Onde o Apache é pronto para executar, o Node é pronto para codificar.

O Node atinge seus objetivos fornecendo servidores altamente escaláveis. Ele não aloca um modelo de encadeamento por conexão, mas usa um modelo processo por conexão, criando somente a memória que é necessária para cada conexão. Ele usa um mecanismo JavaScript extremamente rápido do Google, o mecanismo V8. Ele usa um projeto direcionado a eventos para manter o código mínimo e fácil de ler. Todos esses fatores levam ao objetivo desejado do Node — é relativamente fácil escrever uma solução altamente escalável.

Tão importante quanto entender o que o Node é, é entender o que ele não é. O Node não é simplesmente uma substituição para o Apache que tornará seu aplicativo PHP da Web mais escalável. Isto não pode estar mais longe da verdade. Neste estágio preliminar da vida do Node, ele tem um potencial limitado para uso por muitos programadores, mas nas situações em que faz sentido usá-lo, ele funciona extremamente bem.

O que se pode esperar do Node no futuro? Essa é, talvez, a pergunta

mais importante a levar deste artigo. Agora que você sabe o que ele faz, procure saber o que ele fará a seguir. No próximo ano, espero que o Node ofereça melhor integração com bibliotecas de suporte de terceiros existentes. No momento, muitos programadores terceiros desenvolveram plugins para o Node, incluindo a adição de suporte ao servidor de arquivos e ao MySQL. Espero que o Node comece a integrar isso na funcionalidade principal.

Eventualmente, também espero que o Node suporte algum tipo de módulo de página dinâmica, permitindo fazer os tipos de coisas em um arquivo HTML que é possível fazer no PHP e JSPs (talvez uma NSP, página de servidor Node). Finalmente, em algum momento, espero um servidor Node pronto para implantação, que você faça o download, instale e simplesmente coloque seus arquivos HTML dentro dele como faria com o Apache ou o Tomcat. Ainda é cedo na vida do Node, mas ele está crescendo rapidamente e poderá, em breve, estar em seu horizonte.

Recursos



Aprender

- A [página inicial do Node.js](#) é o ponto inicial para saber mais sobre o aplicativo.
- Navegue na [página da API do Node.js](#). Note que a sintaxe poderá mudar de um release para o outro, portanto verifique a versão baixada com relação à API que está verificando.
- Veja o [Jaxer](#), a primeira grande tentativa de criar um ambiente JavaScript do lado do servidor.
- Leia sobre [Aptana Jaxer](#) no artigo do developerWorks de Michael Galpin.
- Fique por dentro dos [eventos técnicos e webcasts do developerWorks](#).
- Fique por dentro dos [developerWorks](#).
- Siga o [DeveloperWorks no Twitter](#).
- Confira conferências, feiras, webcasts e outros [eventos](#) em todo o mundo que são do interesse dos desenvolvedores de software livre.
- Visite a [zona de software livre](#) do developerWorks para obter instruções extensas, ferramentas e atualizações de projeto para ajudá-lo a desenvolver com tecnologias de software livre e usá-las com produtos da IBM.
- Saiba sobre tecnologias e funções de produto IBM e de software livre

com as [Demos on demand do developerWorks](#).

Obter produtos e tecnologias

- Faça o download do [Node.js](#) e do [Python](#), que também será necessário.
- Consulte a lista em constante evolução de [pacotes de software livre](#) na Wikipédia.
- Inove seu próximo projeto de desenvolvimento de software com a [Versão de teste do software IBM](#), disponível para download.

NEWSLETTER
Fique por dentro de todas as novidades, eventos, cursos e muito mais

ENVIAR

Discutir

- [Participe do fórum de discussão](#).
- Ajude a desenvolver o grupo [Software livre do mundo real](#) na comunidade do developerWorks.

*



Artigo originalmente publicado em *IBM developerWorks*

<http://www.ibm.com/developerworks/br/library/os-nodejs/index.html>

Autor: Michael Abernethy, Programador Freelancer. Trabalha como Gerente de Desenvolvimento de Produto para a Optimal Auctions, uma companhia de software de leilões. Seu foco hoje em dia está em Rich Internet Applications e em torná-los ao mesmo tempo mais complexos e mais simples.



developerWorks Brasil em [JavaScript](#), [Redes e Servidores](#)
segunda-feira, 5 de setembro de 2011

0
SHARES



leia agora

Instalando o ModSecurity no Debian + Apache2



Luiz Cezar
em [Linux](#)





Dê Sua Opinião

O seu endereço de e-mail não será publicado. Campos obrigatórios são marcados com *

NEWSLETTER

Fique por dentro de todas as novidades, eventos, cursos e muito mais

seu nome

seu e-mail

ENVIAR

ENVI

35 comentários em “O que exatamente é o Node.js?”

**Flavio Ferreira**

5 de setembro de 2011 às 11:32

Caraca, explanou bem !! Muitas dúvidas se foram.
Aguardo próximos posts.

[RESPONDER](#)**Flavio Ferreira**

5 de setembro de 2011 às 11:32

Caraca, explanou bem !! Muitas dúvidas se foram.
Aguardo próximos posts.

[RESPONDER](#)**Jean Nascimento**

5 de setembro de 2011 às 12:43

Show de bola eu ja tinh pensado em fazer um artigo aqui sobre o Node, mas o seu ficou bem completo com crtza mais do q
meu ficaria =p
Parabéns.

[RESPONDER](#)**Bruno Thiago Leite Agutoli**

5 de setembro de 2011 às 13:16

Muito bom!

[RESPONDER](#)**Bruno Thiago Leite Agutoli**

5 de setembro de 2011 às 13:17

Muito bom!

[RESPONDER](#)**Davi Oliveira**

5 de setembro de 2011 às 17:00

Excelente artigo sobre o Node. Valeu mesmo!
Obs.: hoje os comentários do iMasters estão com eco.

[RESPONDER](#)**Marlos Carmo**

6 de setembro de 2011 às 6:57

Muito bom o artigo! Fica a sugestão para fazer um neste mesmo conceito falando do NoSQL!

[RESPONDER](#)

6 de setembro de 2011 às 9:31

Show de bola seu artigo.

Fiquei na dúvida em duas coisas: é possível acessar banco de dados com o Node, mas retornar grandes volumes de dados é não é adequado, entendi certo? Você falou que ele não projetado para criar páginas dinâmicas, mas você deu um exemplo de loop, porque exatamente não seria indicado para páginas dinâmicas, uma vez que JS o que mais tem é loop?

Mais uma vez parabéns pelo artigo.

[RESPONDER](#)**Marcelo Pires**

6 de setembro de 2011 às 13:09

Cara existem diversos módulos para o node para acessar MySQL, PostGreeSQL, Mongo e etc. Na página do projeto estão listados os diversos módulos disponíveis.

Eu já estou usando em um projeto e recomendo muito.

[RESPONDER](#)**Marcelo Pires**

6 de setembro de 2011 às 13:10

PS. PostGreSQL

[RESPONDER](#)

NEWSLETTER
Fique por dentro de todas as novidades, eventos, cursos e muito mais

ENVIAR

**Vinícius Faria**

6 de setembro de 2011 às 12:27

Muito bom!!!
Parabéns!!!

RESPONDER

**Cleiton França**

7 de setembro de 2011 às 17:17

Muito bom, espero ansioso o dia em que teremos um servidor puro sangue Node, sei lá, se

RESPONDER

NEWSLETTER

Fique por dentro de todas as novidades, eventos, cursos e muito mais

ENVIAR

**Euler**

15 de setembro de 2011 às 12:56

O Node poderia ser uma opção para desenvolver um sistema php que se comporte semelhante ao Data Push? Ou seja, dado: real time?

RESPONDER

**uNDERsCORE**

3 de outubro de 2011 às 18:13

Creio que a parte “Para o que ele não serve” está errada. O NodeJS já fornece linguagens de templates, o que podemos equi com o JSP por exemplo e há muito tempo.

O NodeJS é perfeito para se trabalhar com banco de dados relacionais também, a grande vantagem do Node é o I/O assíncro tudo construído para Node é assim, incluindo os drivers de bancos de dados relacionais, o que torna natural usar assim:

```
db.query("select * from table", function(err,data){
  if(!err){
    app.render(JSON.stringify(data));
  }
});
```

RESPONDER

**Will**

11 de fevereiro de 2014 às 11:03

Exato. E sem falar que as requisições e manipulações de dados em banco relacionais podem ser usadas de forma síncrona e assíncrona, tornando possível fazer as mesmas rotinas que qualquer outra linguagem, com a diferença que o NODE apresenta vantagem de poder fazer assíncrono se quiser.

RESPONDER

**Rogers**

9 de outubro de 2012 às 21:34

Tentei executar o código mas não tive sucesso. Configurei as variáveis de ambiente e usei a url “http://localhost/?number=27” O Node retornou o seguinte erro no console:

```
var input = param.number;
```

^

ReferenceError: param is not defined
at Server. (C:\Users\Rogers\Desktop\teste.js:31:18)
at Server.EventEmitter.emit (events.js:96:17)
at HTTPParser.parser.onIncoming (http.js:1806:12)
at HTTPParser.parserOnHeadersComplete [as onHeadersComplete] (http.js:111:23)
at Socket.socket.ondata (http.js:1703:22)
at TCP.onread (net.js:403:27)

E o navegador foi redirecionado para a seguinte página: "http://www.localhost.com.br"

[RESPONDER](#)

NEWSLETTER
Fique por dentro de todas as novidades, eventos, cursos e muito mais

ENVIAR

**Guilherme Santos**

25 de julho de 2014 às 10:09

Ao executar está dando o erro param not defined, conforme relatado pelo Roger em comentário anterior.

Para corrigir, basta acertar o nome do objeto na linha 31 do script

de

```
var input = param.number;
```

para

```
var input = params.number;
```

[RESPONDER](#)

Pingback: Node.js – JavaScript no Servidor | Erko Bridee
Pingback: Node.js – JavaScript no Servidor [visão geral] | JavascriptBrasil
Pingback: Node.js – JavaScript no Servidor [visão geral] | JavaScript Brasil

**Rafael Moreira**

5 de junho de 2013 às 10:46

Para quem quiser saber mais sobre o assunto, nós aqui no Brasil temos nossa própria comunidade em português e listada na páginas de comunidades do nodejs. Basta acessar: <http://nodebr.com>

[RESPONDER](#)**Adler**

12 de junho de 2013 às 10:41

Está com erro no trecho

```
var params = url.parse(request.url, true).query;
```

```
var input = param.number;
```

basta trocar o "var input = param.number;" por "var input = params.number;"

[RESPONDER](#)**Caio Ribeiro Pereira**

26 de agosto de 2013 às 9:14

Dois blogs brasileiros que recomendo acessar, pois na minha opinião falam muito sobre Node.js:

Nodebr – <http://nodebr.com>

Underground WebDev – <http://udgwebdev.com>

[RESPONDER](#)

**Will**

11 de fevereiro de 2014 às 10:59

O NODE já implementa o JADE ou o EJS, no projeto EXPRESS, onde se torna possível facilmente utilizar páginas dinâmicas exemplo: dentro da página que será renderizada para o cliente. Fora que o EXPRESS já implementa uma estrutura básica para utilizar o MVC e o pattern para rotas de requisição.

RESPONDER

**bruno**

26 de fevereiro de 2014 às 17:08

É recomendável utilizar node.js em uma plataforma de ecommerce?

RESPONDER

NEWSLETTER×

Fique por dentro de todas as novidades, eventos, cursos e muito mais

ENVIAR

**Thiago Tecedort**

15 de abril de 2014 às 13:03

Muiiito bom o post.

RESPONDER

**Felipe Santos**

13 de agosto de 2014 às 11:56

Excelente artigo, eu tinha muitas duvidas em relação ao node que foram sanadas neste post.

RESPONDER

**André Andrade**

26 de agosto de 2014 às 15:52

Concordo em partes do que foi descrito aqui. Porém, node é muito mais do que aparenta ser pelo texto desse artigo. A começar que para quem não sabe nem java, nem node, nem php o node é de longe o mais rápido para se fazer um HelloWorld funcionar (o server dele é muito simples, extremamente funcional, possui tratamento para dependência de pacotes). No meu ponto de vista ele não substitui java se considerarmos tudo que java é capaz, mas substitui o PHP em todos os casos (ou seja, toda aplicação que faz sentido ser feita em PHP, faz sentido ser feita em Node). Além disso:

- Se conecta a N bancos, inclusive relacionais;
- PHP não ocupa a mesma memória como um Java ocupa (Em nosso DataCenter, não subimos nada em java com pelo menos 1GB de RAM);
- Já temos templates de Node.js, que chega até a ser melhor a nível de arquitetura do que a construção de uma JSP (scriptlet);
- Node não foi feito só para expor serviços, inclusive eu mesmo não utilizaria o mesmo para este propósito (faria em java... em REST, usando jersey);
- Com Node se constrói uma aplicação completa, organizada e performática;
- Node chega a ser mais rápido que PHP (Java então, nem se fala);
- A nível de produtividade, desenvolver com Node é mais rápido que Java ou PHP (óbvio, desde que o programador tenha aprendido a linguagem, que é uma das mais fáceis que já vi);
- Deployar uma aplicação em Node é MUITO simples (muito mais simples que em PHP ou Java). Desenvolver então, nem se fala; use o nodemon e terá redeploy automático enquanto desenvolve;
- Para não ficar tendencioso... tenho 34 anos, sou especialista em Java (16 anos de conhecimento), já programei bastante em (8 anos atrás), CGI/PERL, Python, já trabalhei com Linux, tenho total conhecimento sobre o assunto, domino SOA, WOA, Application Servers (tomcat, jboss, weblogic, websphere), trabalho com tecnologia de ponta (na área de arquitetura corporativa; uma telecom) e já desenvolvi em node.js para conhecer a linguagem (com uso de templates, mysql, mongodb, etc... no Open

e no meu desktop).

O objetivo do meu texto foi animar a turma que quer aprender Node.js. Não irão se arrepender... mãos a obra... é TOP!!!
Temos até uma plataforma conhecida chamada Ghost que é um sistema concorrente do famoso WordPress (não sei qual o m
sei que o Ghost é bom).

Abs!

RESPONDER



Kurosaki

21 de maio de 2015 às 12:54

Java é tão escalável quanto node.js, tendo a diferença que aplicações Java EE não s
feita, que você já explicou. Soluções para web-real time em java são as bibliotecas\fr
e PlayFramework, onde a cada requisição não é iniciada uma thread diferente como j
conexões.Se preferir implementar um servidor altamente escalável em java na “mão”
o java 4 e foi adicionado mais recursos no java 7.

Em questão de simplicidade, eu recomendo o playframework, que é fácil de aprender (tem versões em java e scala).

RESPONDER



Josias

27 de setembro de 2014 às 23:02

Realmente ajudou pouco pq é a mesma coisa que li em outros sites dita com outras palavras.



RESPONDER



Horleilson

1 de novembro de 2015 às 22:37

Não estou sendo exagerado mais esse é o melhor artigo que eu já li em toda internet parabéns mesmo de verdade estou
impressionado :)

RESPONDER



Enoque

27 de novembro de 2015 às 0:05

Parabéns pelo post!! Excelente explicação.

RESPONDER



Vagner Mello

12 de fevereiro de 2016 às 21:13

Excelente!!! Muito bem explicado. Tirou minhas dúvidas sobre o assunto ;)

RESPONDER



Douglas Fernandes

17 de fevereiro de 2016 às 10:21

Parabéns!
Excelente explicação.

Simples e objetiva.

RESPONDER



William Crudeli Junior

4 de maio de 2016 às 15:15

Perfeito explicou bem claro

RESPONDER

Este projeto é mantido e patrocinado pelas empresas:



Hospedado por:



Desenvolvimento

Agile
Ajax
Análise de Dados
CakePHP
CSS
Front End
HTML
Java
JavaScript
PHP
Python
Ruby

Design

3ds max
Acessibilidade
Arquitetura de Informação
Design Responsivo
Games
Usabilidade
User Experience

Banco de dados

Interbase
MongoDB
MySQL
Oracle
PostgreSQL
SQL Server

Infra e Cloud
Cloud Computing
Linux
Microsoft Azure
Segurança
Site Blindado

Marketing Digital

Conteúdo Digital
E-commerce
E-mail Marketing
Mercado
Publicidade Online
Redes Sociais
Tendências

Mobile
Android
iPhone & iPad

Agenda

Fórum
7Masters

Cursos Online

InterCon
Revista iMasters



[Sobre o iMasters](#) [Política de Privacidade](#) [Fale conosco](#) [iMasters Expert \(english blog\)](#)

