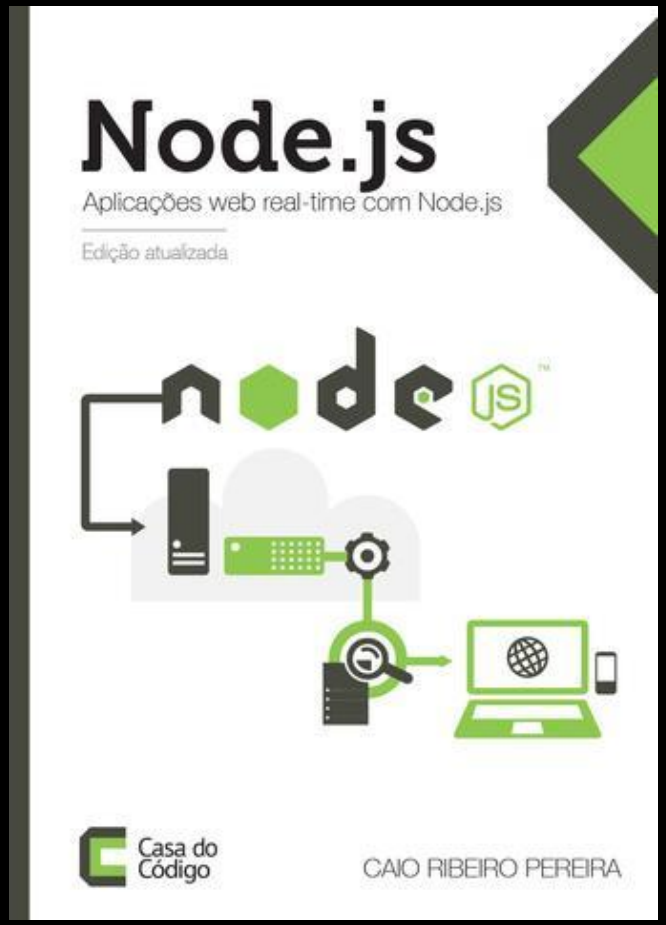
Como fazer um chat online usando Node.js e WebSocket

Alexandre Velloso Pinheiro Filho

2018

Minha motivação para essa palestra



O que é Node.JS?

- Node.js é um ambiente de desenvolvimento *open source*.
- Node.js é gratis.
- Node.js roda em diversas plataformas (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js usa JavaScript no servidor

- Fonte: https://www.w3schools.com/nodejs/nodejs_intro.asp

O que Node.JS pode fazer?

- Node.js pode gerar páginas com conteúdo dinâmico.
- Node.js pode criar, abrir, ler, escrever, deletar e fechar arquivos no servidor.
- Node.js pode coletar dados de um form.
- Node.js pode adicionar, deletar, e modificar o seu banco de dados.

- Fonte: https://www.w3schools.com/nodejs/nodejs_intro.asp

Palavras complicadas que veremos nessa apresentação

- Resposta assíncrona
- Protocolo http
- Api Rest
- Programação orientada a eventos
- Socket.io

Node.js é assíncrono

- Beleza, mas que diabos é isso?
  - Node.js não trava o servidor com uma requisição http, ou seja, se um cliente entra no site e demora muito tempo para carregar uma página, ou faz uma consulta muito custosa no banco de dados ele não interfere em outros usuários.
  - Cada requisição é tratada de forma independente

Protocolo http

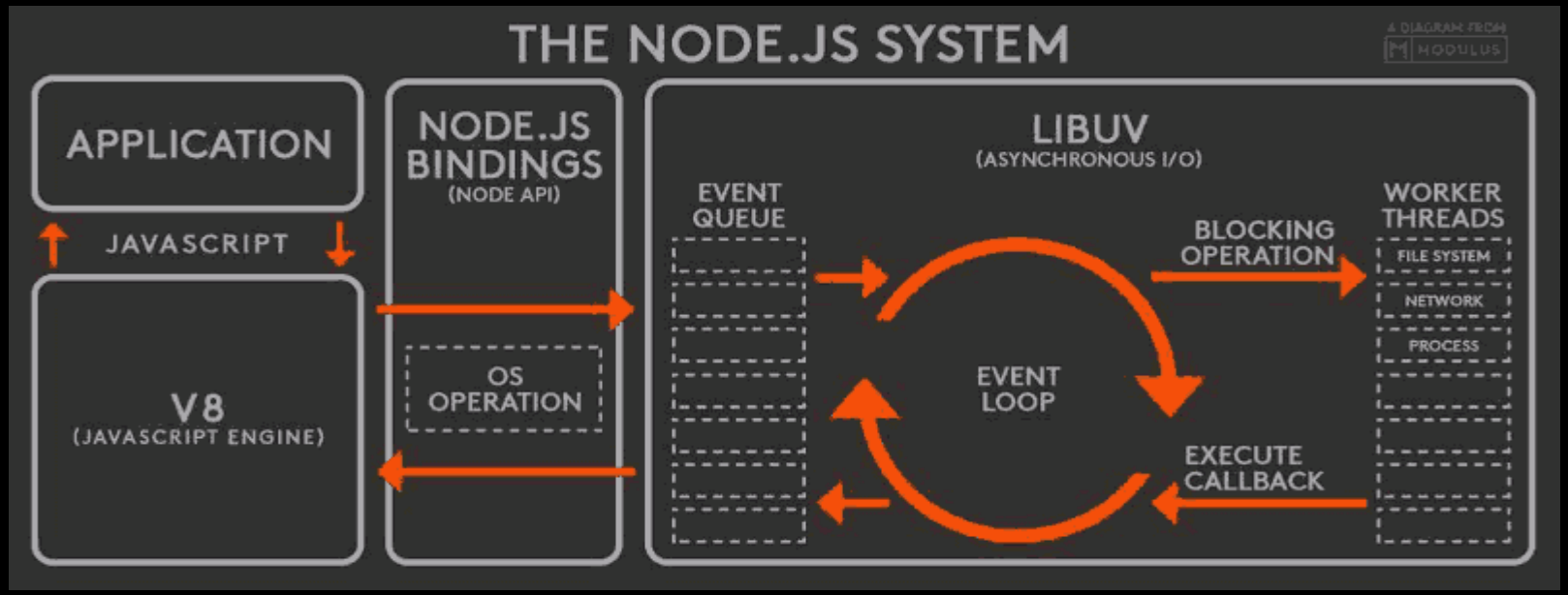
- O protocolo HTTP significa(Hypertext Transfer Protocol), ou Protocolo de Transferência de Hipertexto em portugues é responsável por transferir dados do cliente para e servidor e vice versa.

Api Rest

- A arquitetura Representational State Transfer (REST), em português Transferência de Estado Representacional é a arquitetura que iremos utilizar para fazer alterações no estado da nossa aplicação.
- Métodos:
  - Get: requisita uma informação do servidor
  - Post: insere uma informação no servidor
  - Update: atualiza uma informação do servidor
  - Delete: deleta uma informação do servidor

Node.js usa programação orientada a eventos

- Meu deus!, eu não sei programação orientada a objetos, e agora eventos?
- Calma, é mais fácil do que parece
- Programação orientada a eventos é simples. Imagine que você quer fazer uma consulta em um servidor web, você não sabe quanto tempo essa requisição vai demorar para ser respondida, então você faz a requisição e espera ela te avisar quando acabar.



Socket.io

- Socket.IO permite comunicação bidirectional em real time baseada em eventos.
- Funciona em todas plataformas, navegador ou dispositivo, focando igualmente em confiabilidade e velocidade.
- Usaremos o Socket.io para estabelecer a comunicação em tempo real com o servidor

- Fonte: https://socket.io/

Mãos a obra!

- Com Node.js é possível fazer um servidor com um consumo eficiente de recursos da forma mais simples possível.
- Crie um arquivo chamado app.js, coloque o seguinte código.
- Abra o cmd ou terminal na pasta do arquivo e digite “npm install express” e depois digite “node app.js”.
- Pronto, agora abra o seu navegador e digite “localhost:3000”. Agora procure o seu ip pelo terminal ou cmd e acesse pelo celular. Ex: “192.168.1.10:3000”

```
const fs = require('fs');
const express = require('express');
var app = express();

app.get('/', function(req,res){
    res.end('<h1>Hello World!</h1>');
});

app.listen(3000, function(){
  console.log('servidor rodando na porta 3000');
});
```

Estrutura do nosso chat

- A aplicação terá apenas 1 página, onde nela o usuário irá colocar o nome e poderá mandar mensagens.
- O nosso chat terá apenas 1 sala, onde todos poderão enviar mensagem.
- A versão final do chat pode ser encontrada aqui:
  - https://github.com/AlexandreVelloso/EscolaDeferias/tree/master/2018-1/Como%20fazer%20um%20chat%20online%20usando%20NodeJS/Codigos/Chat

Criar a pagina html

- Crie o arquivo index.html e coloque o seguinte código:

```
<!doctype html>
<html>
  <head>
    <title>Socket.IO chat</title>
    <style>
      * { margin: 0; padding: 0; box-sizing: border-box; }
      body { font: 13px Helvetica, Arial; }
      form { background: #000; padding: 3px; position: fixed; bottom: 0; width: 100%; }
      #m { border: 0; padding: 10px; width: 77%; margin-right: .5%; }
      #label-nome { color: #fff; }
      #nome { border: 0; padding: 10px; width: 10%; margin-right: .5%;}
      form button { width: 9%; background: rgb(130, 224, 255); border: none; padding: 10px; }
      #messages { list-style-type: none; margin: 0; padding: 0; }
      #messages li { padding: 5px 10px; }
      #messages li:nth-child(odd) { background: #eee; }
      #messages { margin-bottom: 40px }
    </style>
  </head>
  <body>
    <ul id="messages"></ul>
    <form action="">
      <label id="label-nome">Nome:</label>
      <input id="nome" />
      <input id="m" autocomplete="off" /><button>Send</button>
    </form>
  </body>
</html>
```

Vamos colocar agora o Socket.io (1/2)

- Primeiro devemos instalar o pacote socket.io executando o comando "npm install socket.io"
- Depois disso, copie o seguinte código para o arquivo app.js

```
var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);
var port = process.env.PORT || 3000;

// rota, ou seja, quando o servidor receber a url localhost:porta/
// ele vai fazer essa acao. Se ele receber uma outra url ele vai dar erro
// pois eu nao tratei ela
app.get('/', function(req, res){
  res.sendFile(__dirname + '/index.html');
});
```

Vamos colocar agora o Socket.io (2/2)

```
io.on('connection', function(socket){
  socket.on('chat message', function(msg){

    const splited = msg.split('|');

    const nome = splited[0];
    const mensagem = splited.slice( 1, splited.length );

    io.emit('chat message', nome+": "+mensagem);
  });
});

http.listen(port, function(){
  console.log('listening on *:' + port);
});
```

Agora vamos alterar a página html e colocar o socket nela (1/2)

- Logo após a tag form adicione o seguinte código:

```
</form>
<script src="https://cdn.socket.io/socket.io-1.2.0.js"></script>
<script src="https://code.jquery.com/jquery-1.11.1.js"></script>
<script>
  $(function () {
    var socket = io();
    $('form').submit(function(){

      if( $('#nome').val() == '' || $('#m').val() == '' ){
        alert('Nome e mensagem não podem ficar vazios');
      }else{

        // mando o nome separado por '|' para o servidor saber o que é nome e o que é mensagem,
        // claro que se a pessoa ter o caractere '|' no nome vai ser um problema
        const mensagem = $('#nome').val() + "|" + $('#m').val();

        // manda a mensagem para o servidor
        socket.emit('chat message', mensagem);
        // limpa a nova mensagem
        $('#m').val('');
        // volta o foco para a caixa de mensagem
        $('#m').focus();
      }
```

Agora vamos alterar a página html e colocar o socket nela (1/2)

```
48
49              // deve se retornar false para o formulario nao ser enviado para o servidor
50              // e o usuario perder todas as mensagens
51              return false;
52            });
53
54            // envento para quando o usuario receber uma mensagem
55            socket.on('chat message', function(msg){
56
57              // adiciona mensagem recebida pelo servidor
58              $('#messages').append($('<li>').text(msg));
59              window.scrollTo(0, document.body.scrollHeight);
60
61            });
62          });
63        </script>
64      </body>
65    </html>
```

E pronto, o seu chat está funcionando!

- Pensem em novas funcionalidades para o chat, e como o Socket.io pode ser utilizado para fazer diversas tarefas.
- Espero que tenham gostado da minha palestra, obrigado.
- Meu github: https://github.com/AlexandreVelloso

- Dúvidas?