

# Making Sense

Introducción a NodeJS

---

# Agenda

- Introduccion a NodeJs.
- Principios de funcionamiento.
- Server basico http.
- Carga de dependencias .
- Rest API con ExpressJS.
- Comunicacion en tiempo real con SocketIO.

# Introducción: Basica

- En palabras simples Node.js es:
  - *JavaScript del lado del servidor .*
- En palabras no tan simples Node.js es:
  - *Un framework para aplicaciones de alto rendimiento, optimizado para entornos de alta concurrencia.*
- Desde el sitio oficial:
  - *'El objetivo de nodeJS es proporcionar una manera fácil de crear programas de red escalables' - (! De nodejs.org)*

# Introducción: Avanzada

- Node.js utiliza un modelo **orientado-a-eventos sin bloqueo de E/S**.
- Hace uso del llamado **bucles-de-eventos** a traves de **callbacks** para implementar el anti-bloqueo de E/S.
- No hay implementación DOM proporcionada por Node.js  
~~`var elemento = document.getElementById("elementId");`~~
- Todo dentro de Node.js se ejecuta en un solo **hilo (single-thread)**.

# Ejemplo de bloqueo de E/S

```
var fs = require('fs');
fs.readFile('/etc/passwd', function(err, buf) {
  console.log(buf.toString()); // Se ejecuta cuando se termine de leer el
});

console.log("Hello!"); //Se ejecuta sin espera
```

<http://runnable.com/VVPEf1ytUnlSSLPA>

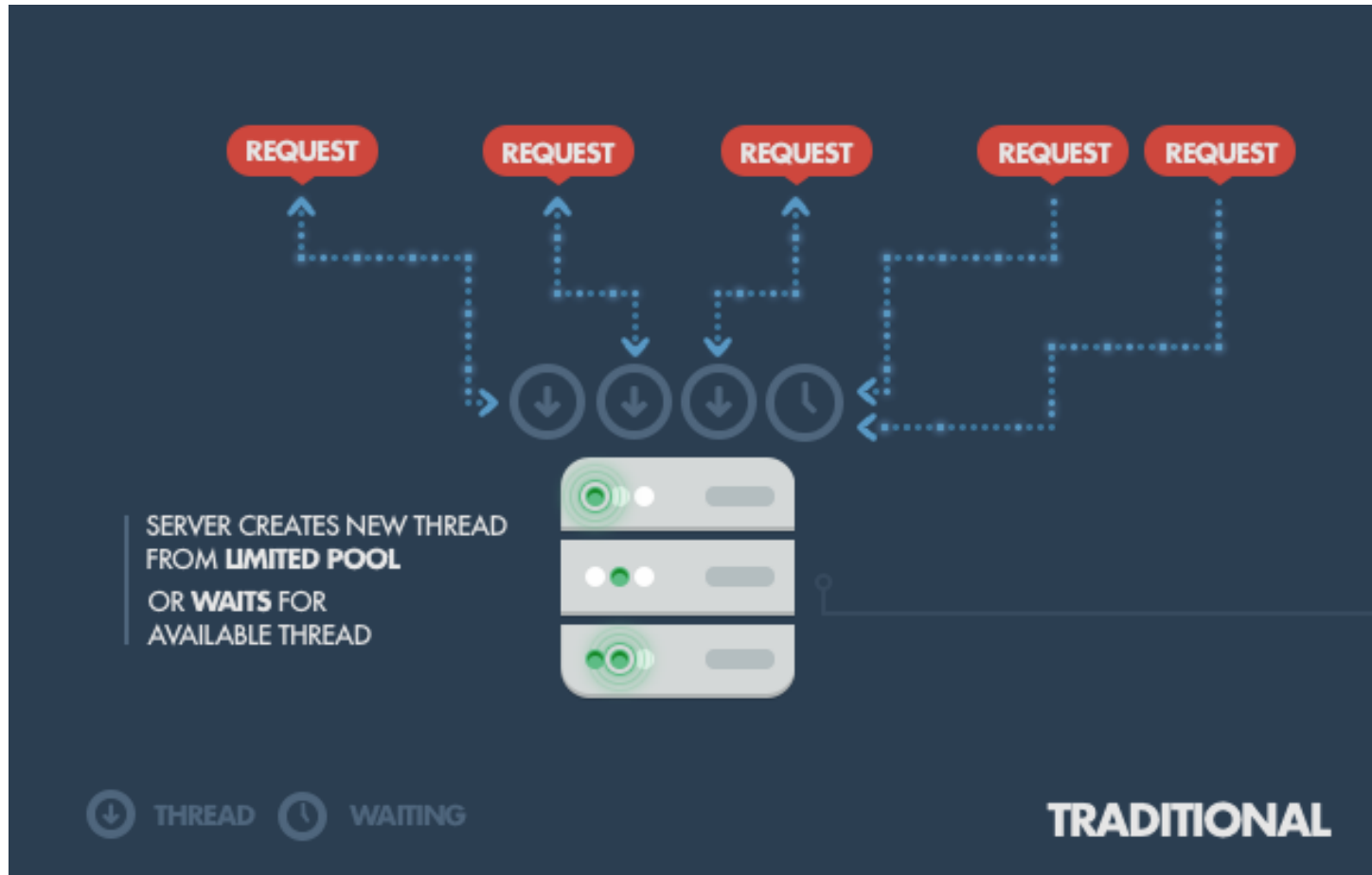
```
var fs = require('fs');
var contents = fs.readFileSync('/etc/passwd').toString();
console.log(contents); //esperamos por el resultado
console.log("Hello!"); //ejecucion bloqueada
```

<http://runnable.com/VVPFREu0oblSplch>

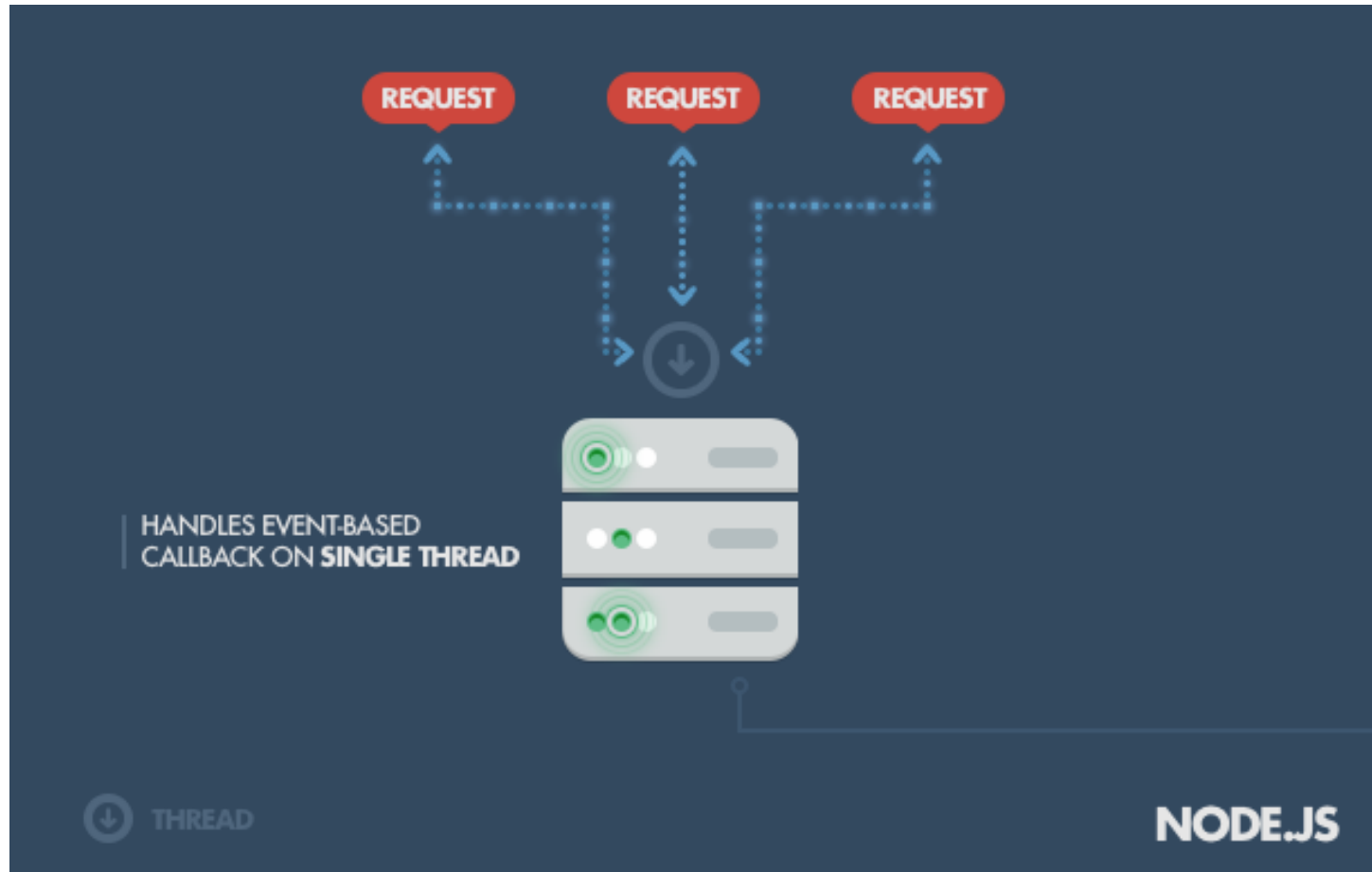
## Para tener en cuenta

- Node.js no es "la" plataforma o "silver bullet" que dominará el mundo del desarrollo web.
- Definitivamente utilizar Node.js para operaciones intensivas de CPU anula casi todas sus ventajas.
- Donde Node.js realmente brilla es en la construcción de aplicaciones de red escalables por su capacidad de manejar un gran número de conexiones simultáneas con un alto rendimiento.

# Funcionamiento de servidores tradicionales

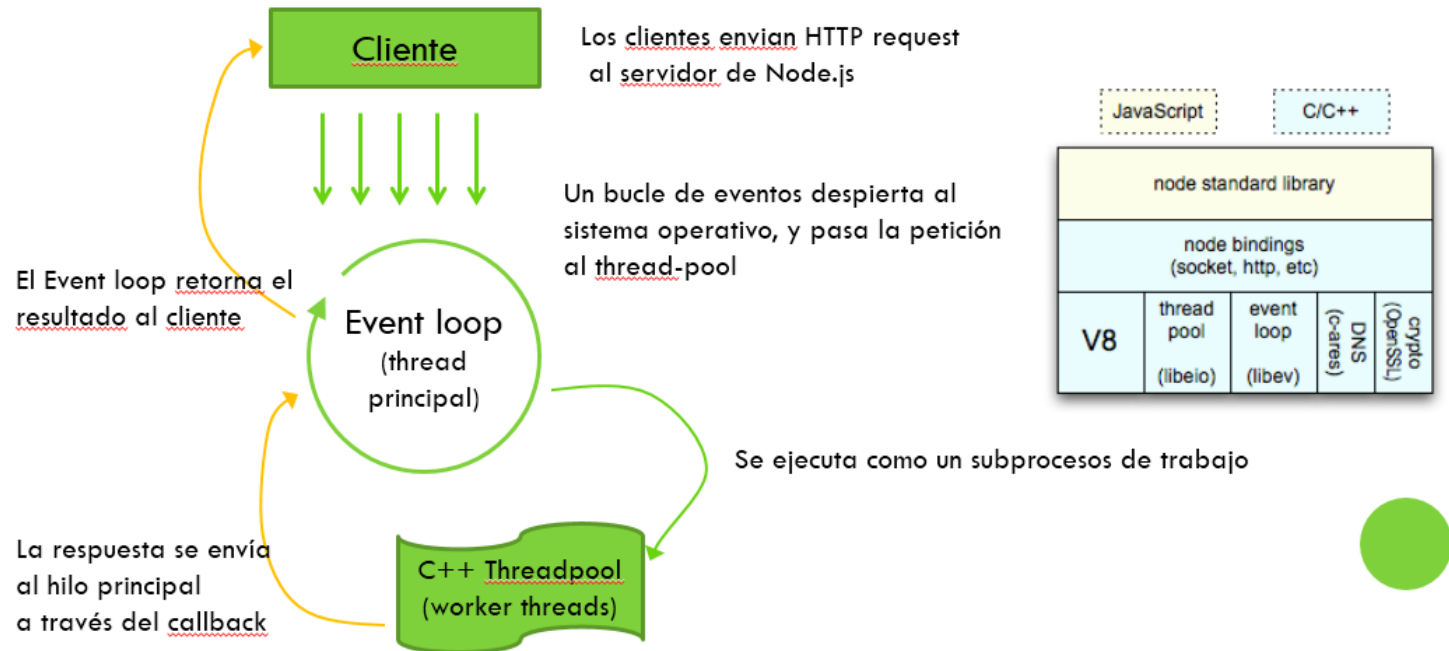


# Funcionamiento de servidor node.js





# Funcionamiento del event-loop



# Server http

```
var http = require("http");  
  
http.createServer(function(req, resp) {  
  resp.writeHead(200, {"Content-Type": "text/html"});  
  resp.write("Hola Mundo");  
  resp.end();  
}).listen(8888);
```

<http://runnable.com/VVSgeKl6sFVrU4vi>

# Echo server http

```
var http = require('http');

http.createServer(function (req, res) {

  req.on('data', function(message) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.write(message);
    res.end();
    console.log(message.toString());
  });

}).listen(80);

console.log('Server listening on port 80');
```

<http://runnable.com/VVFQF8tXtq94AQ-c>

---

# Carga de dependencias

NodeJS - utiliza la implementacion de carga  
modulos basado en CommonJS

1. El codigo usa "require" para incluir los modulos o dependencias
2. Los modules usan "exports" para hacer los modulos disponibles

# El modulo mas simple

```
// hello.js  
console.log('Hello World');
```

```
// app.js  
require('hello.js');
```

<http://runnable.com/VVEzbb7OyS1zCc9B>

# Exportando una funcion anonima

```
// bar.js
module.exports = function () {
  console.log('bar!');
}
```

```
// app.js
var bar = require('./bar.js');
bar();
```

<http://runnable.com/VVE29EGoVsR1j4bQ>

# Exportando un prototype

```
// doo.js
var Doo = function () {};

Doo.prototype.log = function () {
  console.log('doo!');
}

module.exports = Doo;
```

```
// app.js
var Doo = require('./doo.js');
var doo = new Doo();
doo.log();
```

<http://runnable.com/VVFKvnNPkDR3to5Q>

# El "Revealing Module Pattern"

```
module.exports = (function() {  
    var text = "Hello world!"  
  
    var private= function() {  
        console.log("private method executed")  
    };  
  
    var public= function() {  
        console.log(text);  
        private();  
    };  
  
    return {  
        public: public  
    };  
  
})();  
module.exports = module;
```

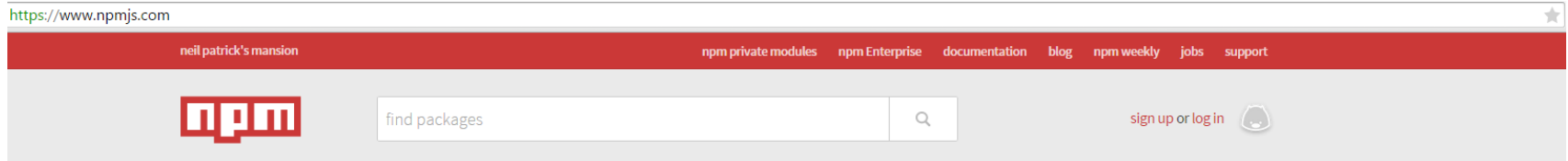
<https://github.com/tfmontague/definitive-module-pattern>

[http://www.w3schools.com/js/js\\_function\\_closures.asp](http://www.w3schools.com/js/js_function_closures.asp)

<http://runnable.com/VVFA0Dj3bI92hDSt>





# El gran repositorio de modulos




npm is the package manager for **javascript**.

 **148,067**  
total packages

 **68,161,658**  
downloads in the last day

 **348,331,576**  
downloads in the last week

 **1,410,242,447**  
downloads in the last month

**packages people 'npm install' a lot**



**browserify**

browser-side require() the node way  
9.0.3 published 3 months ago by substack

express

**express**

Fast, unopinionated, minimalist web framework  
4.12.0 published 3 months ago by dougwilson



**pm2**

Production process manager for Node.JS applications w...  
0.12.6 published 3 months ago by jshkurti



**grunt-cli**

The grunt command line interface.  
0.1.13 published a year ago by tkellen



**npm**

a package manager for JavaScript  
2.6.0 published 3 months ago by othiyim23



**karma**

Spectacular Test Runner for JavaScript.  
0.12.31 published 4 months ago by karmarunnerbot

# Express

Framework Web Rápido,  
De Mente Abierta,  
minimalista para **Node.js**

```
$ npm install express --save
```

Documentación de Express disponible en otros idiomas: [Inglés](#), [Japonés](#), [Ruso](#), [Chino](#).

## Aplicaciones Web

Express es un framework web mínimo y flexible para Node.js que proporciona un conjunto robusto de características para aplicaciones web y móviles.

## APIs

Con una gran variedad de métodos de utilidad HTTP y middleware a su disposición, la creación de una potente API es rápido y fácil.

## Rendimiento

Express proporciona una capa delgada de características fundamentales de aplicaciones web, sin ocultar las características del Node que usted sabe y ama.

# Hello World con ExpressJs

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

app.get('/', function(req, res){
  res.send('Hello World');
});

var server = app.listen(80, function () {
  console.log('listening port: ', server.address().port);
});
```

[http://runnable.com/VVOs8P6u0BlOWP3\\_](http://runnable.com/VVOs8P6u0BlOWP3_)

# Hello World con ExpressJs

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

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.post('/', function (req, res) {
  res.send('Got a POST request');
});

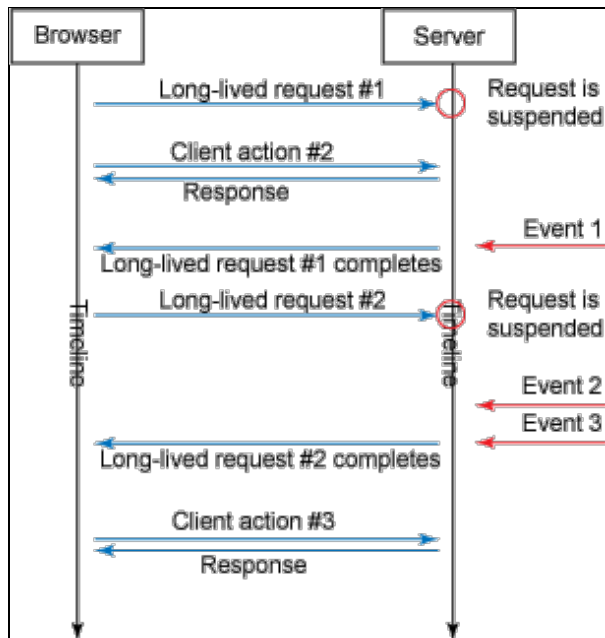
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user');
});

app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user');
});

var server = app.listen(80, function () {
  console.log('listening port: ', server.address().port);
});
```

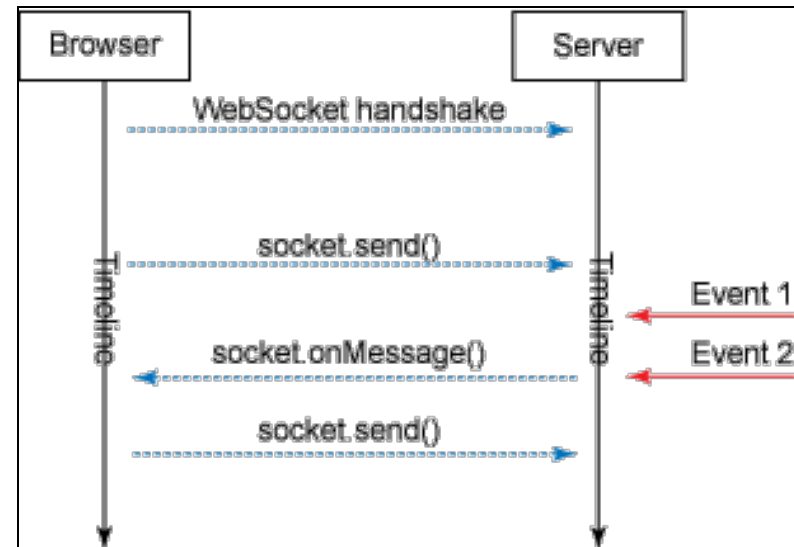
[http://runnable.com/VVOzY\\_6u0BlOWP7l](http://runnable.com/VVOzY_6u0BlOWP7l)

# Problema: ¿como armar un chat web?



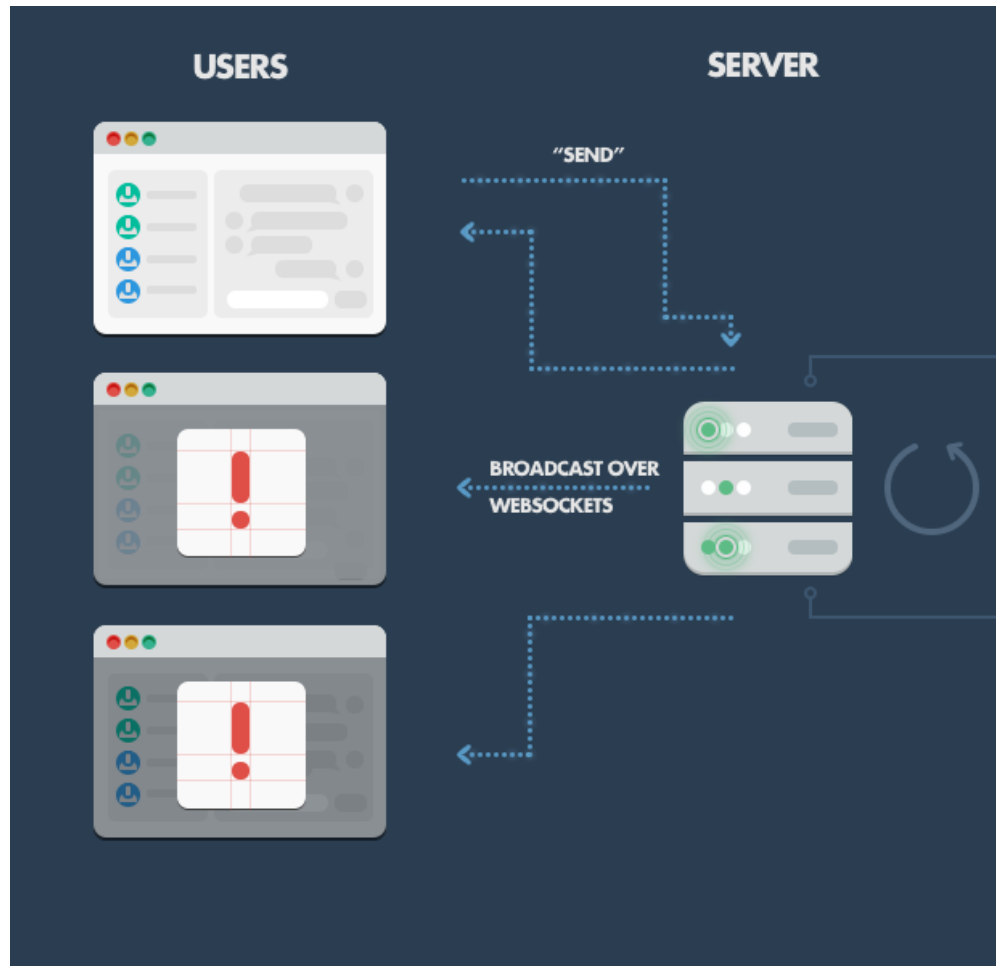
Long-polling

VS



Web Sockets

# Solucion: node y web sockets



# Paso1 - Armar un server simple

```
var app = require('express')();  
var http = require('http').Server(app);  
  
app.get('/', function(req, res){  
  res.send('<h1>Hello world</h1>');  
});  
  
http.listen(80, function(){  
  console.log('listening on *:80');  
});
```

<http://runnable.com/VVNOp2LgYUNDB2EI>

## Paso2: Agregar una pagina como recurso

```
app.get('/', function(req, res){  
  res.sendFile(__dirname + '/index.html');  
});
```

```
<!doctype html>  
<html>  
  <head>  
    <title>Socket.IO chat</title>  
  </head>  
  <body>  
    <ul id="messages"></ul>  
    <form action="">  
      <input id="m" autocomplete="off" /><button>Send</button>  
    </form>  
  </body>  
</html>
```

<http://runnable.com/VVNOp2LgYUNDB2EI>



## Paso3 - Agregar websockets

```
var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);

http.listen(80, function(){
  console.log('listening on *:80');
});

app.get('/', function(req, res){
  res.sendFile(__dirname + '/index.html');
});

io.on('connection', function(socket){
  console.log('a user connected');

  socket.on('disconnect', function(){
    console.log('user disconnected');
  });

  socket.on('chat message', function(msg){
    console.log('message: ' + msg);
    io.emit('chat message', msg);
  });
});
```

[http://runnable.com/VVNqcqgehr-pGk6\\_P](http://runnable.com/VVNqcqgehr-pGk6_P)

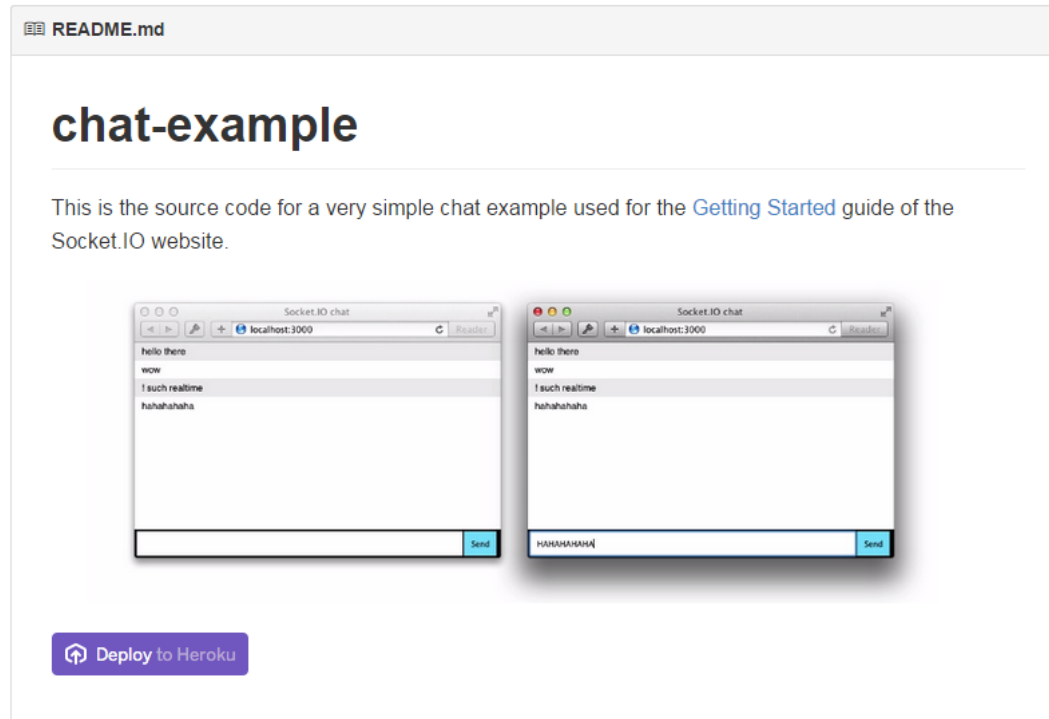
## Paso4: Actualizar nuestra pagina

```
<!doctype html>
<html>
  <head>
    <title>Socket.IO chat</title>
  </head>
  <body>
    <ul id="messages"></ul>
    <form action="">
      <input id="message" autocomplete="off" /><button>Send</button>
    </form>
    <script src="http://cdn.socket.io/socket.io-1.2.0.js"></script>
    <script src="http://code.jquery.com/jquery-1.11.1.js"></script>
    <script>
      var socket = io();
      $('form').submit(function(){
        socket.emit('chat message', $('#message').val());
        $('#message').val('');
        return false;
      });
      socket.on('chat message', function(msg){
        $('#messages').append($('- ').text(msg));
      });
    </script>
  </body>
</html>

```

[http://runnable.com/VVNqcqgehr-pGk6\\_P](http://runnable.com/VVNqcqgehr-pGk6_P)

# Armamos nuestro server en un host



<https://github.com/makingsensetraining/chat-example>

# Challenge

- Transmitir un mensaje a los usuarios conectados cuando alguien se conecta o se desconecta
- Añadir soporte para apodos
- No enviar el mismo mensaje al usuario que lo envió a sí mismo
- Mostrar quién está en línea
- Añadir mensajería privado

A word cloud featuring the phrase "Thank you" in numerous languages. The word "Gracias" is the largest and most central. Other prominent words include "Merci", "Hvala", and "Danke". Smaller words are scattered around, including "Grazie", "Arigato", "Tack", "Merci", "Hvala", "Danke", "Grazie", "Arigato", "Tack", "Merci", "Hvala", "Danke", "Grazie", "Arigato", "Tack", "Merci", "Hvala", "Danke".