Socket.io

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Socket.IO enables real-time bidirectional event-based communication. It works on every platform, browser or device, focusing equally on reliability and speed. Socket.IO is built on top of the WebSockets API (Client side) and Node.js. It is one of the most depended upon library on **npm** (Node Package Manager).

Audience

This tutorial has been created for anyone who has a basic knowledge of HTML, Javascript and Node.js work. After completing this tutorial, the reader will be able to build moderately complex real-time websites, back-ends for mobile applications and push notification systems.

Prerequisites

The reader should have a basic knowledge of HTML, JavaScript and Node.js. If the readers are not acquainted with these, we will suggest them to go through these tutorials first. We will be using Express to ease creating servers; it is not a prerequisite though.

Copyright & Disclaimer

Table of Contents

1. Socket.IO – Overview

Socket.IO is a JavaScript library for **real-time web applications**. It enables real-time, bi-directional communication between web clients and servers. It has two parts: a **client-side library** that runs in the browser, and a **server-side library** for node.js. Both components have an identical API.

Real-time Applications

A real-time application (RTA) is an application that functions within a period that the user senses as immediate or current.

Some examples of real-time applications are:

- **Instant messengers:** Chat apps like Whatsapp, Facebook Messenger, etc. You need not refresh your app/website to receive new messages.
- **Push Notifications:** When someone tags you in a picture on Facebook, you receive a notification instantly.
- **Collaboration Applications:** Apps like google docs, which allow multiple people to update same documents simultaneously and apply changes to all people's instances.
- **Online Gaming:** Games like Counter Strike, Call of Duty, etc., are also some examples of real-time applications.

Why Socket.IO?

Writing a real-time application with popular web applications stacks like LAMP (PHP) has traditionally been very hard. It involves polling the server for changes, keeping track of timestamps, and it is a lot slower than it should be.

Sockets have traditionally been the solution around which most real-time systems are architected, providing a bi-directional communication channel between a client and a server. This means that the server can push messages to clients. Whenever an event occurs, the idea is that the server will get it and push it to the concerned connected clients.

Socket.IO is quite popular, it is used by **Microsoft Office**, **Yammer**, **Zendesk**, **Trello**, and numerous other organizations to build robust real-time systems. It one of the most powerful **JavaScript frameworks** on **GitHub**, and most depended-upon NPM (Node Package Manager) module. Socket.IO also has a huge community, which means finding help is quite easy.

ExpressJS

We will be using express to build the web server that Socket.IO will work with. Any other node-server-side framework or even node HTTP server can be used. However, ExpressJS makes it easy to define routes and other things. To read more about express and get a basic idea about it, head to – ExpressJS tutorial.

2. Socket.IO – Environment

To get started with developing using the **Socket.IO**, you need to have **Node** and **npm (node package manager)** installed. If you do not have these, head over to **Node setup** to install node on your local system. Confirm that node and npm are installed by running the following commands in your terminal.

```
node --version
npm --version
```

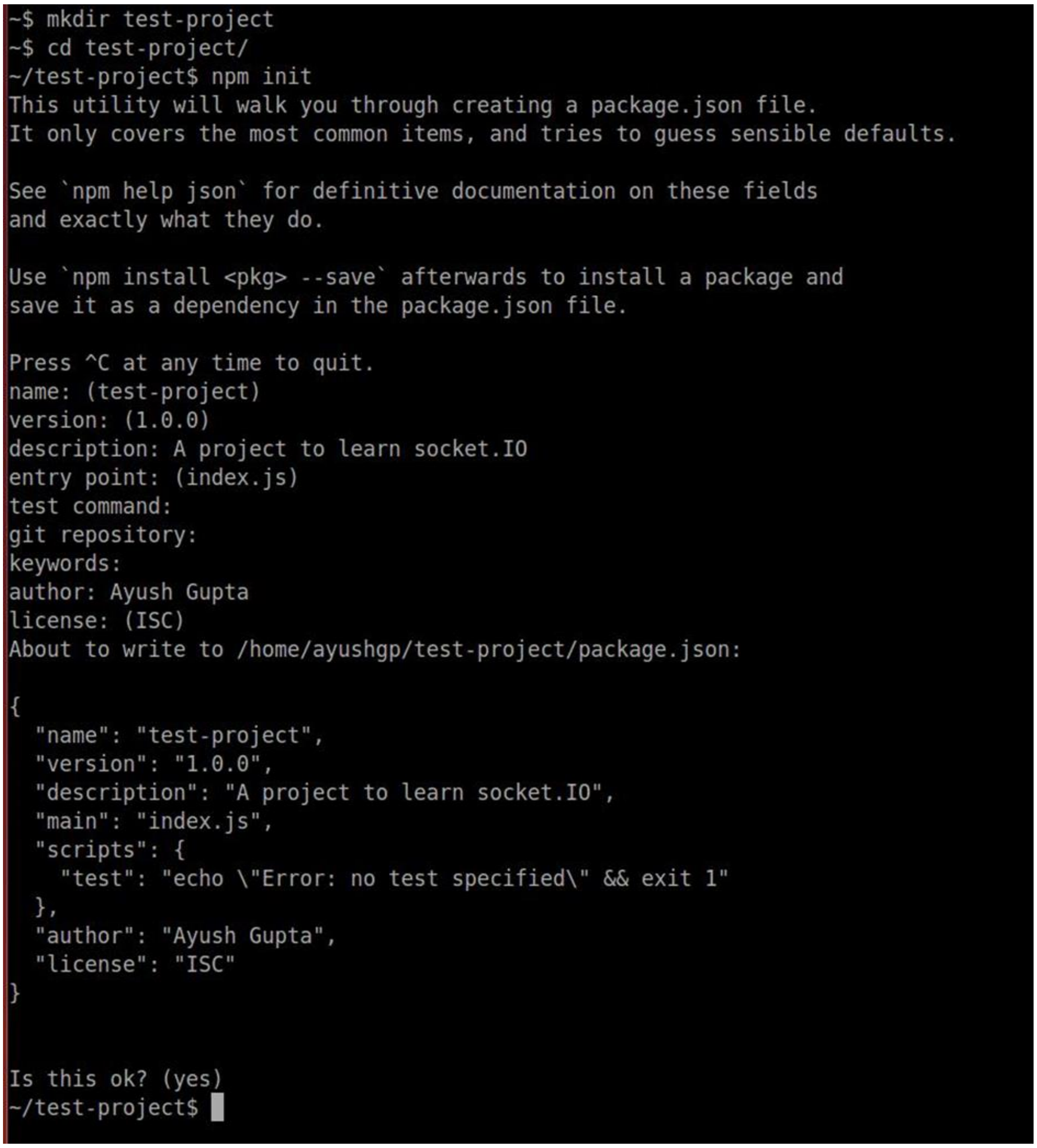
You should get an output similar to:

```
v5.0.0
3.5.2
```

Open your terminal and enter the following in your terminal to create a new folder and enter the following commands:

```
$ mkdir test-project
$ cd test-proect
$ npm init
```

It will ask you some questions; answer them in the following way:



```
~$ mkdir test-project
~$ cd test-project/
~/test-project$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (test-project)
version: (1.0.0)
description: A project to learn socket.IO
entry point: (index.js)
test command:
git repository:
keywords:
author: Ayush Gupta
license: (ISC)
About to write to /home/ayushgp/test-project/package.json:

{
  "name": "test-project",
  "version": "1.0.0",
  "description": "A project to learn socket.IO",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Ayush Gupta",
  "license": "ISC"
}


Is this ok? (yes)
~/test-project$
```

This will create a **'package.json node.js'** configuration file. Now we need to install **Express** and **Socket.IO**. To install these and save them to **package.json** file, enter the following command in your terminal, into the project directory:

```
npm install --save express socket.io
```

One final thing is that we should keep restarting the server. When we make changes, we will need a tool called **nodemon**. To install nodemon, open your terminal and enter the following command:

```
npm install -g nodemon
```

Whenever you need to start the server, instead of using the **node app.js** use, **nodemon app.js**. This will ensure that you do not need to restart the server whenever you change a file. It speeds up the development process.

Now, we have our development environment set up. Let us now get started with developing real-time applications with Socket.IO.