Pusher Js with Node Js

Ritik Khatri Jul 8, 2019 · 6 min read

source

Pusher sits as a real-time layer between your servers and your clients. Pusher maintains persistent connections to the clients — over WebSocket if possible and falling back to HTTP-based connectivity — so that as soon as your servers have new data that they want to push to the clients they can do, instantly via Pusher.

We are going to create a realtime application with help of pusherJs and NodeJs as backend.

Setting up your NPM project

Goto the project directory, open your **terminal**, type commands below:

```
# this will create a directory named nodeJs-pusherJS
mkdir nodeJs-pusherJS

# this will change your directory to your newly created directory
cd nodeJs-pusherJS

# this will initialize your NPM project with default rules
npm init -y
```

We have now successfully created a NPM project, It's time to add the some dependencies for our project.

🔰 Adding Dependencies

Now open your terminal, type commands below:

```
npm install --save express body-parser pusher
```

- express - It is to create the web server for the REST API that is going to use.
- body-parser - It is a middleware for parsing the body of the request.
- pusher - It is to publish changes in realtime.

Now in terminal ,type command below:

```
npm install --save-dev nodemon
```

nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.

Add scripts in package.json file so that we do not have to restart the server after the changes .

```
"scripts": {
    "start": "node server.js",
```

```
    "dev": "nodemon server.js"
}
```

The final `package.json` file will be similar to:

```
{

 "name": "pusherjs",
 "version": "1.0.0",
 "description": "",
 "main": "server.js",
 "scripts": {
 "test": "echo \"Error: no test specified\" && exit 1",
 "start": "node server.js",
 "dev": "nodemon server.js"
 },
 "author": "",
 "license": "ISC",
 "dependencies": {
  "body-parser": "^1.19.0",
  "express": "^4.17.1",
  "pusher": "^2.2.1"
 },
 "devDependencies": {
 "nodemon": "^1.19.1"
 }
 }
```

🎛 Setting up Project Directory

Now we'll setup our project structure to have some initial files. Our project directory will be looking similar to :

```
|____node_modules
|____public
|______index.html
|______client.js
|____package-lock.json
|____package.json
|____server.js
```

As per this project structure, you'll need to create the folders and files, if they don't exist. Don't worry if you are not getting it, once it is done, you'll get it.

First of all, we need to create a server.js file which can hold our server side code.

So create a server.js in root directory and put the code shown to create a basic express server.

```
const express = require('express');
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static('public'));

app.get('/', (request, response) => {

});

app.listen(3000, () => {
  console.log('Express intro running on localhost:3000');
});
```

Then add express.static middleware to server static part of server.

```
app.use(express.static('public'));
```

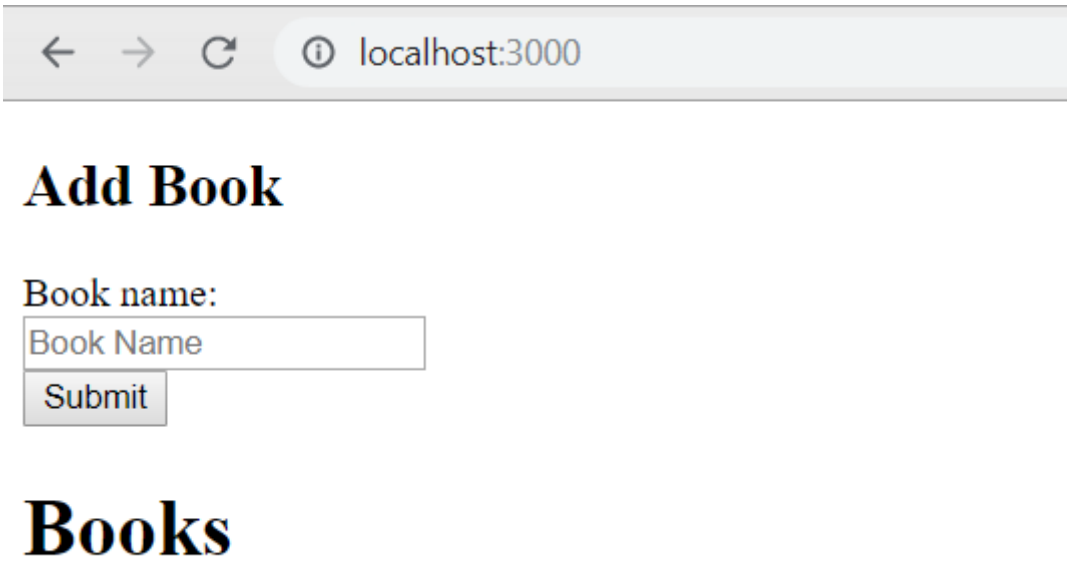
Now create a new folder in root directory called public and then create a new file called index.html and put the code shown below

```
<!DOCTYPE html>
<html>
<head>
   <title>PusherJs Application</title>
</head>
<body>
<h2>Add Book </h2>
   <form id="book-form" >
   Book name: <br>
   <input type="text" id="book-name" name="bookname"
placeholder="Book Name">
<br>
<input type="submit" value="Submit">
</form>
<h1> Books <h1>
<div class="book-list" id="books-list"></div>
<script src='./client.js'></script>
</body>
</html>
```

Now start server by writing this code in the terminal

```
npm run dev
```

Then head on over to localhost:3000 in your browser, and you should see:



🔰 Create Pusher Credentials

If you haven't already, create a pusher account at *Pusher*.

Then, go to your dashboard and create a Channels app, choosing a name, the cluster closest to your location, and optionally, React as the frontend tech and Node.js as the backend tech:

This will give you some sample code to get started:

Save your app id, key, secret and cluster values. We'll need them later.

Use Pusher In Server and Client Side

Now add this code in server.js to create Pusher instance

```
var Pusher = require('pusher');

//create a instance of pusher using your credentials
var pusher = new Pusher({
  appId: API_ID,
  key: API_KEY,
  secret: API_SECRET,
  cluster: 'ap2',
  encrypted: true
});
```

Now also add pusher in client side by adding this script in html file

```
<script src="https://js.pusher.com/4.4/pusher.min.js"></script>
```

After this create a client.js file in public directory and write code to create instance of pusher in client side. As we already added script of pusher in html so we can use in client.js

```
//  To make Channels a bit more chatty about what is coming in via
// the socket connection
Pusher.logToConsole = true;

var pusher = new Pusher(api_key, {
      cluster: 'ap2',
    });
```

After this create a new post request route “/post” in server to handle a request of adding a new book. Before that create a empty array called books to store names of books.

```
//create a variable to store books name
var books = [] ;

//create a new route to update books name
app.post('/post',(req,res)=>{
//add new book from body to array of books
   books.push(req.body.book);
//trigger  a event in a channel with json object
   pusher.trigger('channel-name', 'event-name', {
       data: books
  });

//send respone for this route
   res.json({
       message : "Book added succesfully"})
  })
```

Whenever this route get call pusher will trigger a event called add in a channel post with a object of books and send a JSON object of message.

Now add a event listener on form submission in client.js. On submission of form this will call route ‘/post’ using fetch api and method will be POST. we will also send the bookname with api in body.

```
var bookForm = document.getElementById("book-form");

//add a event Listener on form submit
bookForm.addEventListener("submit",addbook);

const addbook =(e)=>{
e.preventDefault();

//store name of new book in a variable
const newBook = document.getElementById('book-name').value;

 //call route '/post' to trigger event with new book
fetch("http://localhost:3000/post",{
    method:"POST",
    headers: {
        Content-Type: 'application/json'
       },
   body :JSON.stringify({
        book : newBook
       })
   })
  .then(res=>res.json())
  .then(json=>{
    console.log(json.message)
   })
}
```

Now on the submission of new book pusher trigger a event with a object containing updated lists of books in a channel called post. So we have to subscribe that channel. To get books from that channel we have to bind that channel with event by channel.bind() and use a callback function to use data attched with event. To check whether data is right or wrong we will console the data.

```
//subscribe the channel to bind events of channel
var channel = pusher.subscribe('channel-name');

//bind the event of the presence channel to get the data attched
//with it

channel.bind('add', function(data) {
       console.log(data.books)
});
```

After this add a script of jquery in the end of our html file so that we can render our books name in class book-list.

```
<script src='https://code.jquery.com/jquery-2.2.1.min.js',
integrity='sha256-gvQgAFzTH6trSrAWoH1iPo9Xc96QxSZ3feW6kem+O00=',
crossorigin='anonymous'></script>
```

Now in the channel.bind() function render the last element of so that whenever a new book will added we can see that under the Books section.

```
//render the data of a event in division using jQuery
var length = data.books.length;
$('#books-list').append($('<li>').html('<b>'+data.books[length-
1]+'</b>'));
```

📌 Preview

[*Source Code*]

To preview the app, Just open your `terminal` in the project directory and type below command-

```
npm run dev
```

If all goes well, the app will fire up at **`localhost:3000`** (your port can be different, for that check your console logs of server).

💚 If you like this article, Please **Shareit** and if you have any queries regarding PusherJs ,NodeJS and this setup, please **Comment** out.

JavaScript Pusher Nodejs Realtime Database


Download on the
App Store


GET IT ON
Google Play