

Javascript (<https://www.sitepoint.com/javascript/>) > April 30, 2018

> By [Michael Wanyoike](https://www.sitepoint.com/author/mwanyoike/) (<https://www.sitepoint.com/author/mwanyoike/>).

Building a WebRTC Video Chat Application with SimpleWebRTC

With the advent of WebRTC and the increasing capacity of browsers to handle peer-to-peer communications in real time, it's easier than ever to build real-time applications. In this tutorial, we'll take a look at SimpleWebRTC (<https://simplewebrtc.com/>), and how it can make our lives easier when implementing WebRTC. Throughout the article, we'll be building a WebRTC video chat app with messaging features.

If you need a bit of a background regarding WebRTC and peer-to-peer communication, I recommend reading The Dawn of WebRTC (<http://www.sitepoint.com/the-dawn-of-webrtc/>), and Introduction to the getUserMedia API (<http://www.sitepoint.com/introduction-getusermedia-api/>).

What is SimpleWebRTC

Before we move on, it's important that we understand the main tool that we'll be using. SimpleWebRTC (<https://github.com/andyet/SimpleWebRTC>) is a JavaScript library that simplifies WebRTC peer-to-peer data, video, and audio calls.

SimpleWebRTC acts as a wrapper around the browser's WebRTC implementation. As you might already know, browser vendors don't exactly agree on a single way of implementing different features, which means that for every browser there's a different implementation for WebRTC. As the developer, you'd have to write different code for every browser you plan to support. SimpleWebRTC acts as the wrapper for that code. The API that it exposes is easy to use and understand, which makes it a really great candidate for implementing cross-browser WebRTC.

Building the WebRTC Video Chat App

Now it's time to get our hands dirty by building the app. We'll build a single page application that runs on top of an Express server.

Please note that you can download the code for this tutorial from our [GitHub repo](https://github.com/sitepoint-editors/simplewebrtc-messenger) (<https://github.com/sitepoint-editors/simplewebrtc-messenger>). To run it, or to follow along at home, you'll need to have Node and npm installed. If you're not familiar with these, or would like some help getting them installed, check out our previous tutorials:

[Install Multiple Versions of Node.js using nvm](https://www.sitepoint.com/quick-tip-multiple-versions-node-nvm/) (<https://www.sitepoint.com/quick-tip-multiple-versions-node-nvm/>).

[A Beginner's Guide to npm — the Node Package Manager](https://www.sitepoint.com/beginners-guide-node-package-manager/)

(<https://www.sitepoint.com/beginners-guide-node-package-manager/>).

You also need a PC or laptop that has a webcam. If not, you'll need to get yourself a USB webcam that you can attach to the top of your monitor. You'll probably need a friend or a second device to test remote connections.

Dependencies

We'll be using the following dependencies to build our project:

[SimpleWebRTC](https://simplewebrtc.com/) (<https://simplewebrtc.com/>), — the WebRTC library

[Semantic UI CSS](https://semantic-ui.com/) (<https://semantic-ui.com/>), — an elegant CSS framework

[jQuery](http://jquery.com/) (<http://jquery.com/>), — used for selecting elements on the page and event handling.

[Handlebars](http://handlebarsjs.com/) (<http://handlebarsjs.com/>), — a JavaScript templating library, which we'll use to generate HTML for the messages

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) ([/legals](#)), and [Privacy Policy](#). ([/privacy-policy](#)). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

Project Setup

Go to your workspace and create a folder `simplewebrtc-messenger`. Open the folder in VSCode or your favorite editor and create the following files and folder structure:

```
simplewebrtc-messenger
├── public
│   ├── images
│   │   └── image.png
│   ├── index.html
│   └── js
│       └── app.js
├── README.md
└── server.js
```

Or, if you prefer, do the same via the command line:

```
mkdir -p simplewebrtc-messenger/public/{images,js}
cd simplewebrtc-messenger
touch public/js/app.js public/index.html .gitignore README.md server.js
```

Open `README.md` and copy the following content:

```
# Simple WebRTC Messenger

A tutorial on building a WebRTC video chat app using SimpleWebRTC.
```

Add the line `node_modules` to the `.gitignore` file if you plan to use a git repository. Generate the `package.json` file using the following command:

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) ([/legals](#)), and [Privacy Policy](#). ([/privacy-policy](#)). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

You should get the following output:

```
(https://www.sitepoint.com/).  
{  
  "name": "simplewebrtc-messenger",  
  "version": "1.0.0",  
  "description": "A tutorial on building a WebRTC video chat app using  
SimpleWebRTC.",  
  "main": "server.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1",  
    "start": "node server.js"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

Now let's install our dependencies:

```
npm install express handlebars jquery semantic-ui-css simplewebrtc
```

As the installation proceeds, copy this code to `server.js`:

Understood

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#), and [Privacy Policy. \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

(<https://www.sitepoint.com/>).

```
const express = require('express');

const app = express();
const port = 3000;

// Set public folder as root
app.use(express.static('public'));

// Provide access to node_modules folder from the client-side
app.use('/scripts', express.static(`${__dirname}/node_modules/`));

// Redirect all traffic to index.html
app.use((req, res) => res.sendFile(`${__dirname}/public/index.html`));

app.listen(port, () => {
  console.info('listening on %d', port);
});
```

The server code is pretty standard. Just read the comments to understand what's going on.

Next, let's set up our `public/index.html` file:

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#), and [Privacy Policy \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

(<https://www.sitepoint.com/>).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="scripts/semantic-ui-css/semantic.min.css">
  <title>SimpleWebRTC Demo</title>
  <style>
    html { margin-top: 20px; }
    #chat-content { height: 180px; overflow-y: scroll; }
  </style>
</head>
<body>
  <!-- Main Content -->
  <div class="ui container">
    <h1 class="ui header">Simple WebRTC Messenger</h1>
    <hr>
  </div>

  <!-- Scripts -->
  <script src="scripts/jquery/dist/jquery.min.js"></script>
  <script src="scripts/semantic-ui-css/semantic.min.js"></script>
  <script src="scripts/handlebars/dist/handlebars.min.js "></script>
  <script src="scripts/simplewebrtc/out/simplewebrtc-with-adapter.bundle.js">
</script>
  <script src="js/app.js"></script>
</body>
</html>
```

Next, let's set up our base client-side JavaScript code. Copy this code to `public/js/app.js`:

```
window.addEventListener('load', () => {
```

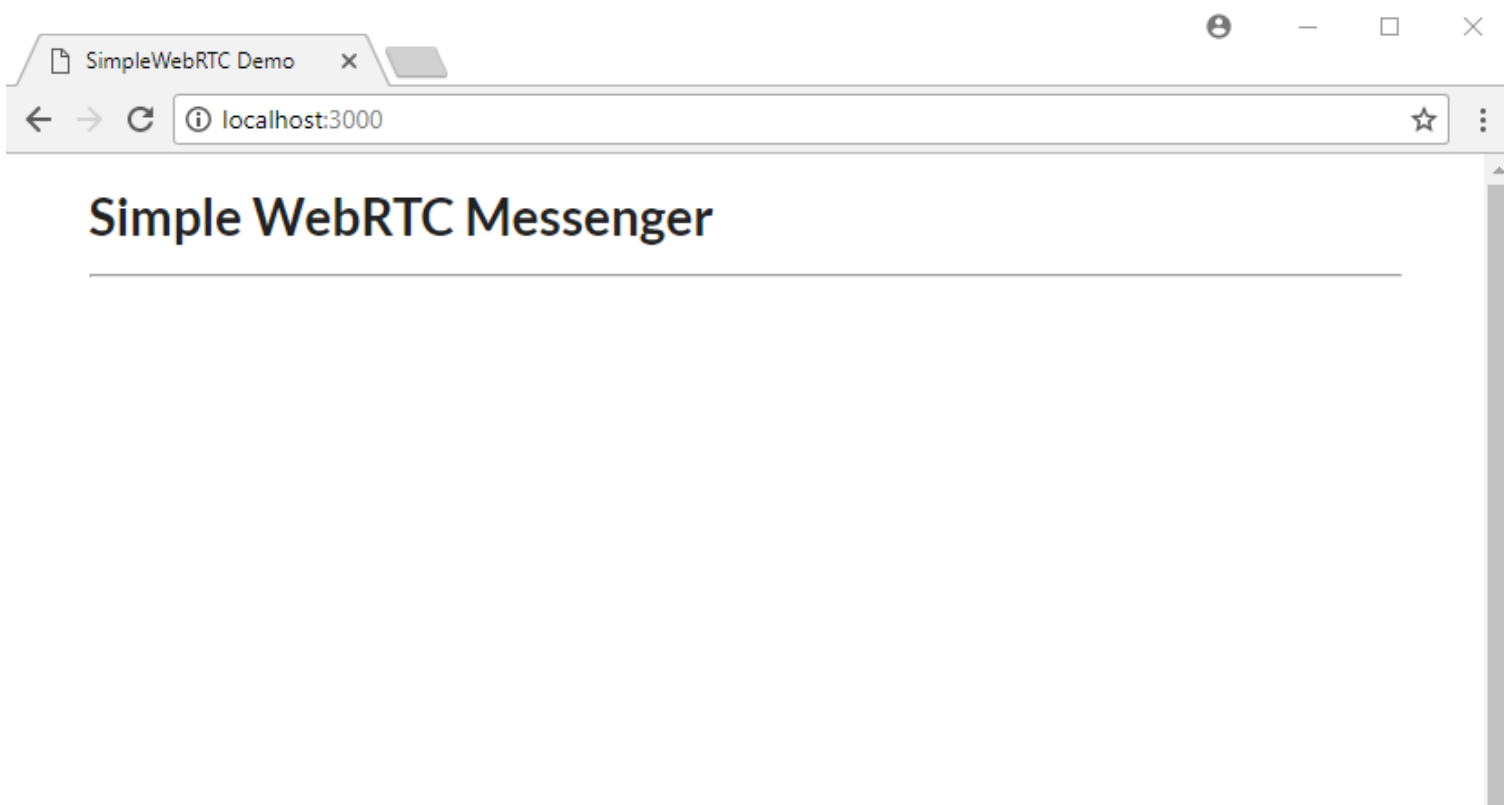
~~// Put all client-side code here~~
We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) ([/legal](#)) and [Privacy Policy](#). ([/privacy-policy](#)). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

Finally, download this [image](https://www.sitepoint.com/) (<https://github.com/sitepoint-editors/simplewebrtc-messenger/blob/master/public/images/image.png>), from our GitHub repository and save it inside the `public/images` folder.

Now we can run our app:

```
npm start
```

Open the URL `localhost:3000` (<http://localhost:3000>), in your browser and you should see the following:



Markup

Let's now work on `public/index.html`. For the sake of simplicity (especially if you're already familiar with Handlebars) you can copy the entire markup code from our [GitHub repository](https://github.com/sitepoint-editors/simplewebrtc-messenger/blob/master/public/index.html) (<https://github.com/sitepoint-editors/simplewebrtc-messenger/blob/master/public/index.html>). Otherwise, let's go through things step by step. First off, copy this code and place it after the `<hr>` tag within the `ui container` div:

(<https://www.sitepoint.com/>).

```
<div class="ui two column stackable grid">
```

```
<!-- Chat Section -->
```

```
<div class="ui ten wide column">
```

```
<div class="ui segment">
```

```
<!-- Chat Room Form -->
```

```
<div class="ui form">
```

```
<div class="fields">
```

```
<div class="field">
```

```
<label>User Name</label>
```

```
<input type="text" placeholder="Enter user name" id="username"
```

```
name="username">
```

```
</div>
```

```
<div class="field">
```

```
<label>Room</label>
```

```
<input type="text" placeholder="Enter room name" id="roomName"
```

```
name="roomName">
```

```
</div>
```

```
</div>
```

```
<br>
```

```
<div class="ui buttons">
```

```
<div id="create-btn" class="ui submit orange button">Create Room</div>
```

```
<div class="or"></div>
```

```
<div id="join-btn" class="ui submit green button">Join Room</div>
```

```
</div>
```

```
</div>
```

```
<!-- Chat Room Messages -->
```

```
<div id="chat"></div>
```

```
</div>
```

```
</div>
```

```
<!-- End of Chat Section -->
```

```
<!-- Local Camera -->
```

```
<div class="ui six wide column">
```

```
<h4 class="ui center aligned header" style="margin:0;">
```

```
Local Camera
```

```
</h4>
```

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) and [Privacy Policy](#). Have questions? Please contact support@sitepoint.com.

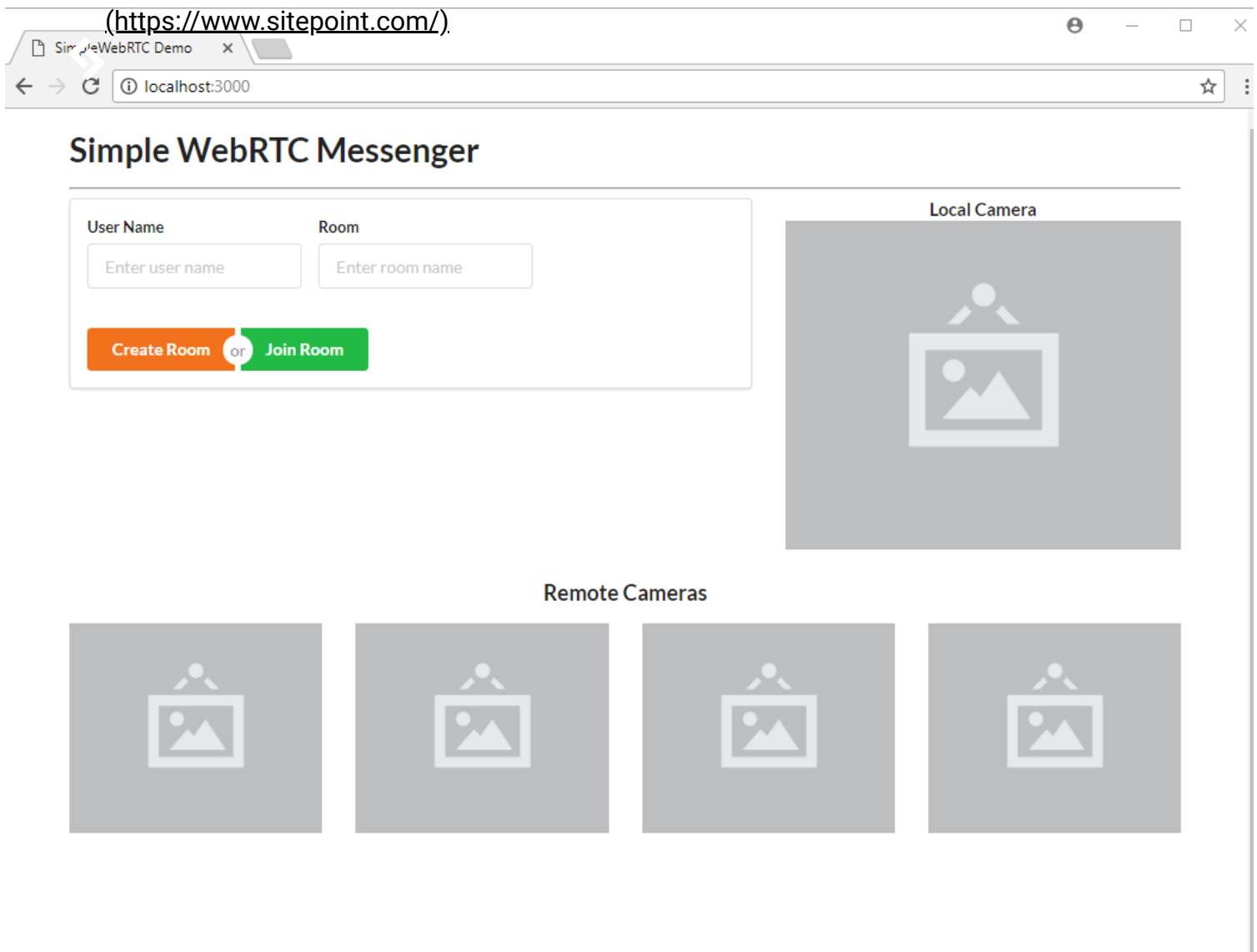
support@sitepoint.com), We're happy to help!

```
</div>
```


</div>[\(https://www.sitepoint.com/\)](https://www.sitepoint.com/).

```
<!-- Remote Cameras -->
<h3 class="ui center aligned header">Remote Cameras</h3>
<div id="remote-videos" class="ui stackable grid">
  <div class="four wide column">
    
  </div>
  <div class="four wide column">
    
  </div>
  <div class="four wide column">
    
  </div>
  <div class="four wide column">
    
  </div>
</div>
```

Go through the markup code and read the comments to understand what each section is for. Also check out the [Semantic UI \(https://www.semantic-ui.com\)](https://www.semantic-ui.com) documentation if you're unfamiliar with the CSS library. Refresh your browser. You should have the following view:



We're using a blank image as a placeholder to indicate where the camera location will stream to on the web page. Take note that this app will be able to support multiple remote connections, provided your internet bandwidth can handle it.

Templates

Now let's add the three Handlebar templates that will make our web page interactive.

Place the following markup right after the **ui container** div (although the location doesn't really matter). We'll start off with the chat container, which simply is made up of:

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

empty chat messages container (to be populated later via JavaScript)

(<https://www.sitepoint.com/>)
input for posting messages.

```
<!-- Chat Template -->
<script id="chat-template" type="text/x-handlebars-template">
  <h3 class="ui orange header">Room ID -> <strong>{{ room }}</strong></h3>
  <hr>
  <div id="chat-content" class="ui feed"> </div>
  <hr>
  <div class="ui form">
    <div class="ui field">
      <label>Post Message</label>
      <textarea id="post-message" name="post-message" rows="1"></textarea>
    </div>
    <div id="post-btn" class="ui primary submit button">Send</div>
  </div>
</script>
```

Next, add the following template, which will be used to display user chat messages:

```
(https://www.sitepoint.com/).
<!-- Chat Content Template -->
<script id="chat-content-template" type="text/x-handlebars-template">
  {{#each messages}}
    <div class="event">
      <div class="label">
        <i class="icon blue user"></i>
      </div>
      <div class="content">
        <div class="summary">
          <a href="#"> {{ username }}</a> posted on
          <div class="date">
            {{ postedOn }}
          </div>
        </div>
        <div class="extra text">
          {{ message }}
        </div>
      </div>
    </div>
  {{/each}}
</script>
```

Finally, add the following template, which will be used to display streams from a remote camera:

```
<!-- Remote Video Template -->
<script id="remote-video-template" type="text/x-handlebars-template">
  <div id="{{ id }}" class="four wide column"></div>
</script>
```

The markup code is hopefully pretty self-explanatory, so let's move on to writing the client-side JavaScript code for our application.

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) ([View our Privacy Policy](#)) ([Privacy Policy](#)). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

Open the file `public/js/app.js` and add this code:

```
(https://www.sitepoint.com/)  
// Chat platform  
const chatTemplate = Handlebars.compile($('#chat-template').html());  
const chatContentTemplate = Handlebars.compile($('#chat-content-  
template').html());  
const chatEl = $('#chat');  
const formEl = $('.form');  
const messages = [];  
let username;  
  
// Local Video  
const localImageEl = $('#local-image');  
const localVideoEl = $('#local-video');  
  
// Remote Videos  
const remoteVideoTemplate = Handlebars.compile($('#remote-video-  
template').html());  
const remoteVideosEl = $('#remote-videos');  
let remoteVideosCount = 0;  
  
// Add validation rules to Create/Join Room Form  
formEl.form({  
  fields: {  
    roomName: 'empty',  
    username: 'empty',  
  },  
});
```

Here we're initializing several elements that we plan to manipulate. We've also added validation rules to the form so that a user can't leave either of the fields blank.

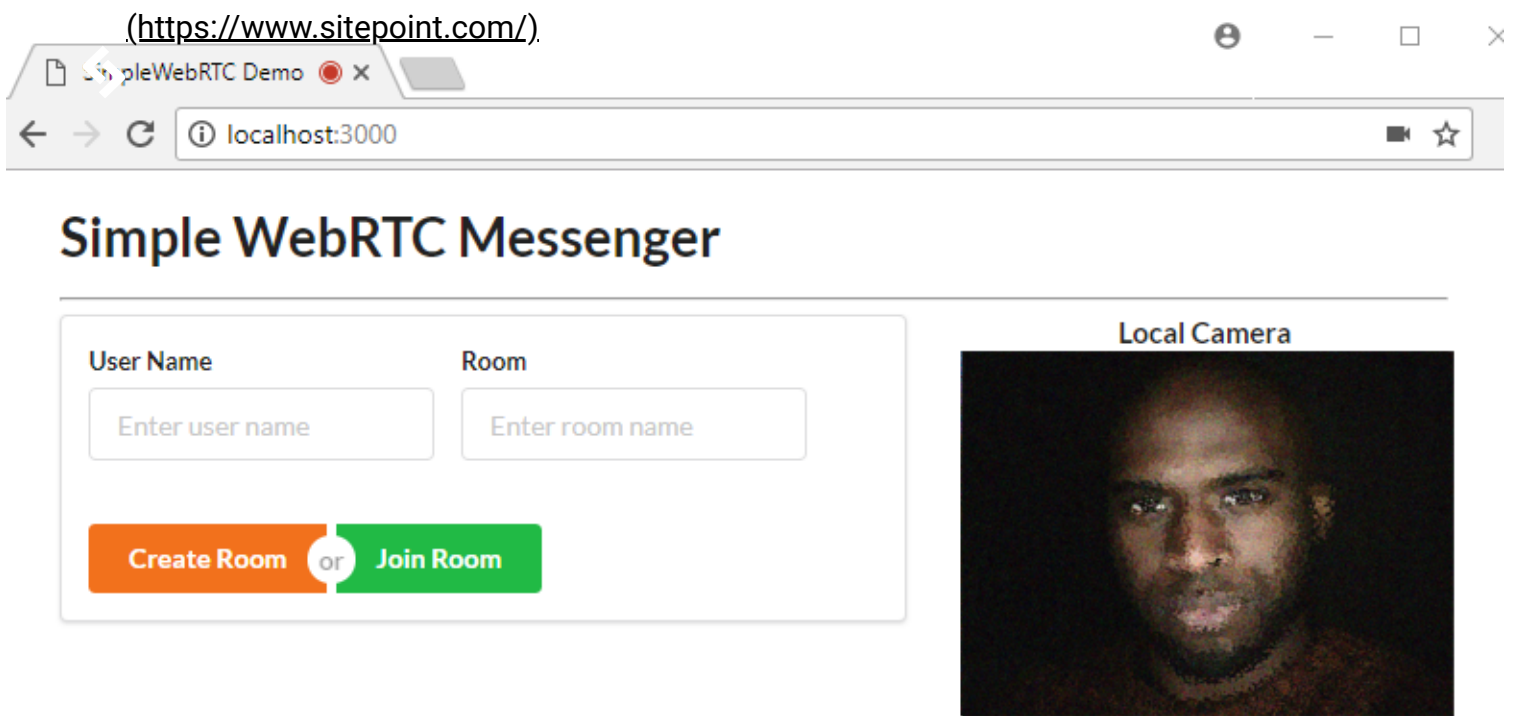
Next, let's initialize our WebRTC code:

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#), and [Privacy Policy. \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

```
(https://www.sitepoint.com/).  
// create our WebRTC connection  
const webrtc = new SimpleWebRTC({  
  // the id/element dom element that will hold "our" video  
  localVideoEl: 'local-video',  
  // the id/element dom element that will hold remote videos  
  remoteVideosEl: 'remote-videos',  
  // immediately ask for camera access  
  autoRequestMedia: true,  
});  
  
// We got access to local camera  
webrtc.on('localStream', () => {  
  localImageEl.hide();  
  localVideoEl.show();  
});
```

Now you know why it's called SimpleWebRTC. That's all we need to do to initialize our WebRTC code. Noticed we haven't even specified any ICE servers or STUN servers. It just works. However, you can use other TURN services such as [Xirsys](https://xirsys.com/simplewebrtc/) (<https://xirsys.com/simplewebrtc/>). You'll need to set up a local [SignalMaster](https://github.com/andyet/signalmaster) (<https://github.com/andyet/signalmaster>), server for handling WebRTC signaling.

Let's do a quick refresh of the web page to confirm the new code is working:



Remote Cameras



The page should request access to your camera and microphone. Just click *Accept* and you should get the above view.

Chat Room Script

Now let's make the form functional. We need to write logic for creating and joining a room. In addition, we need to write additional logic for displaying the chat room. We'll use the **chat-room-template** for this. Let's start by attaching click handlers to the form's buttons:

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

```

    (https://www.sitepoint.com/).
$(`.submit').on('click', (event) => {
  if (!formEl.form('is valid')) {
    return false;
  }
  username = $('#username').val();
  const roomName = $('#roomName').val().toLowerCase();
  if (event.target.id === 'create-btn') {
    createRoom(roomName);
  } else {
    joinRoom(roomName);
  }
  return false;
});

```

Next, we need to declare the **createRoom** and **joinRoom** functions. Place the following code before the click handler code:

```

// Register new Chat Room
const createRoom = (roomName) => {
  console.info(`Creating new room: ${roomName}`);
  webrtc.createRoom(roomName, (err, name) => {
    showChatRoom(name);
    postMessage(`${username} created chatroom`);
  });
};

// Join existing Chat Room
const joinRoom = (roomName) => {
  console.log(`Joining Room: ${roomName}`);
  webrtc.joinRoom(roomName);
  showChatRoom(roomName);
  postMessage(`${username} joined chatroom`);
};

```

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) ([Legal](#)) and [Privacy Policy](#) ([privacy-policy](#)). Have questions? Please contact [Support@WebRTC.com](mailto:support@webRTC.com) (<mailto:support@sitepoint.com>), we're happy to help!

Creating or joining a room is as simple as that! Just use **[SimpleWebRTC's createRoom and joinRoom methods](https://github.com/andyet/SimpleWebRTC#methods)** (<https://github.com/andyet/SimpleWebRTC#methods>).

(<https://www.sitepoint.com/>)

You may also have noticed that we have **showChatroom** and **postMessage** functions that we haven't defined yet. Let's do that now by inserting the following code before the calling code:

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#), and [Privacy Policy. \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

```

    (https://www.sitepoint.com/).
// Post Local Message
const postMessage = (message) => {
  const chatMessage = {
    username,
    message,
    postedOn: new Date().toLocaleString('en-GB'),
  };
  // Send to all peers
  webRTC.sendToAll('chat', chatMessage);
  // Update messages locally
  messages.push(chatMessage);
  $('#post-message').val('');
  updateChatMessages();
};

```

```

// Display Chat Interface
const showChatRoom = (room) => {
  // Hide form
  formEl.hide();
  const html = chatTemplate({ room });
  chatEl.html(html);
  const postForm = $('#form');
  // Post Message Validation Rules
  postForm.form({
    message: 'empty',
  });
  $('#post-btn').on('click', () => {
    const message = $('#post-message').val();
    postMessage(message);
  });
  $('#post-message').on('keyup', (event) => {
    if (event.keyCode === 13) {
      const message = $('#post-message').val();
      postMessage(message);
    }
  });
};

```

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) ([/legals](#)) and [Privacy Policy](#). ([/privacy-policy](#)). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

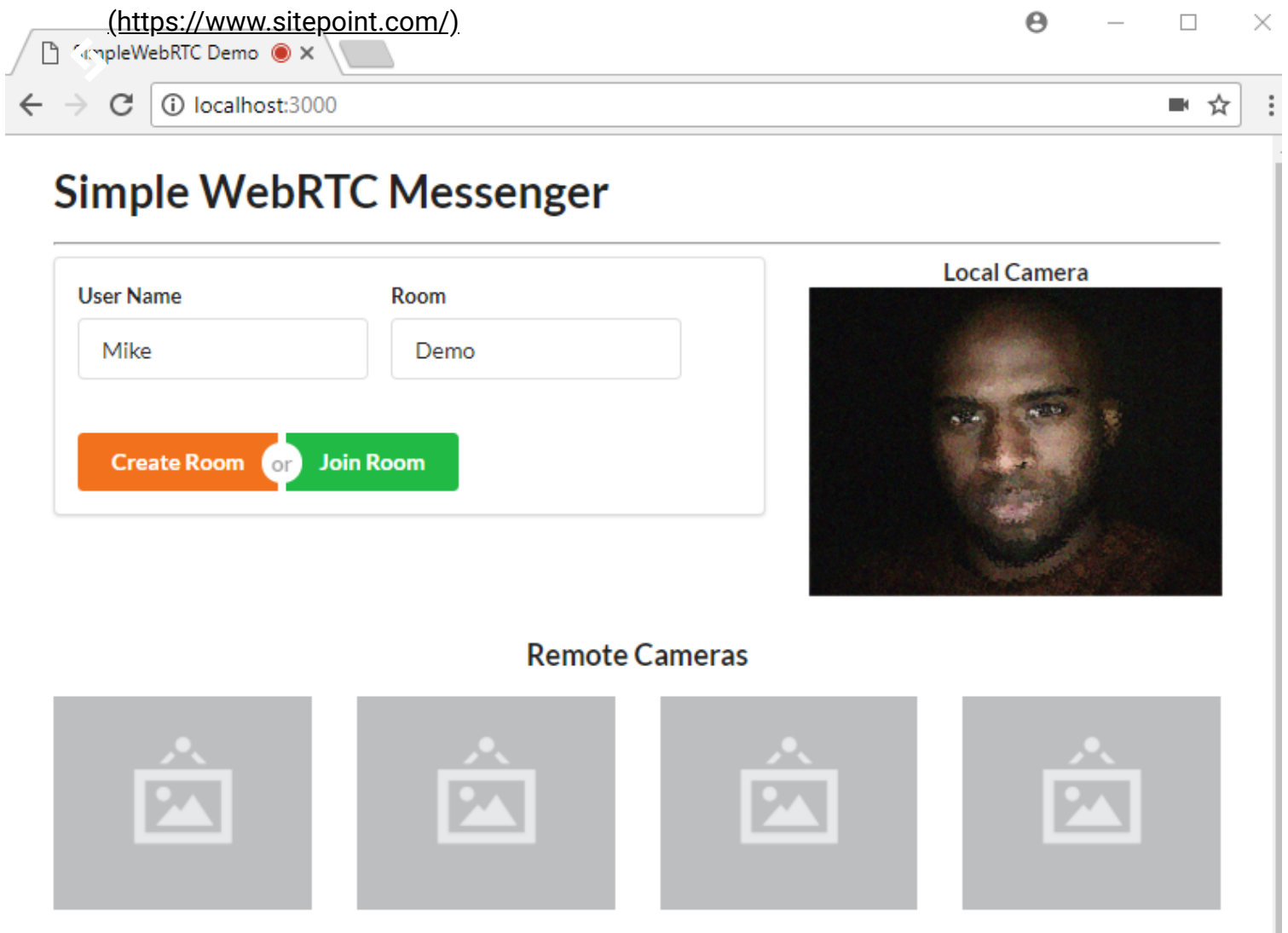
Take some time to go through the code to understand the logic. You'll soon come across another function we haven't declared, **updateChatMessages**. Let's add it now:

```
(https://www.sitepoint.com/).  
// Update Chat Messages  
const updateChatMessages = () => {  
  const html = chatContentTemplate({ messages });  
  const chatContentEl = $('#chat-content');  
  chatContentEl.html(html);  
  // automatically scroll downwards  
  const scrollHeight = chatContentEl.prop('scrollHeight');  
  chatContentEl.animate({ scrollTop: scrollHeight }, 'slow');  
};
```

The purpose of this function is simply to update the Chat UI with new messages. We need one more function that accepts messages from remote users. Add the following function to **app.js**:

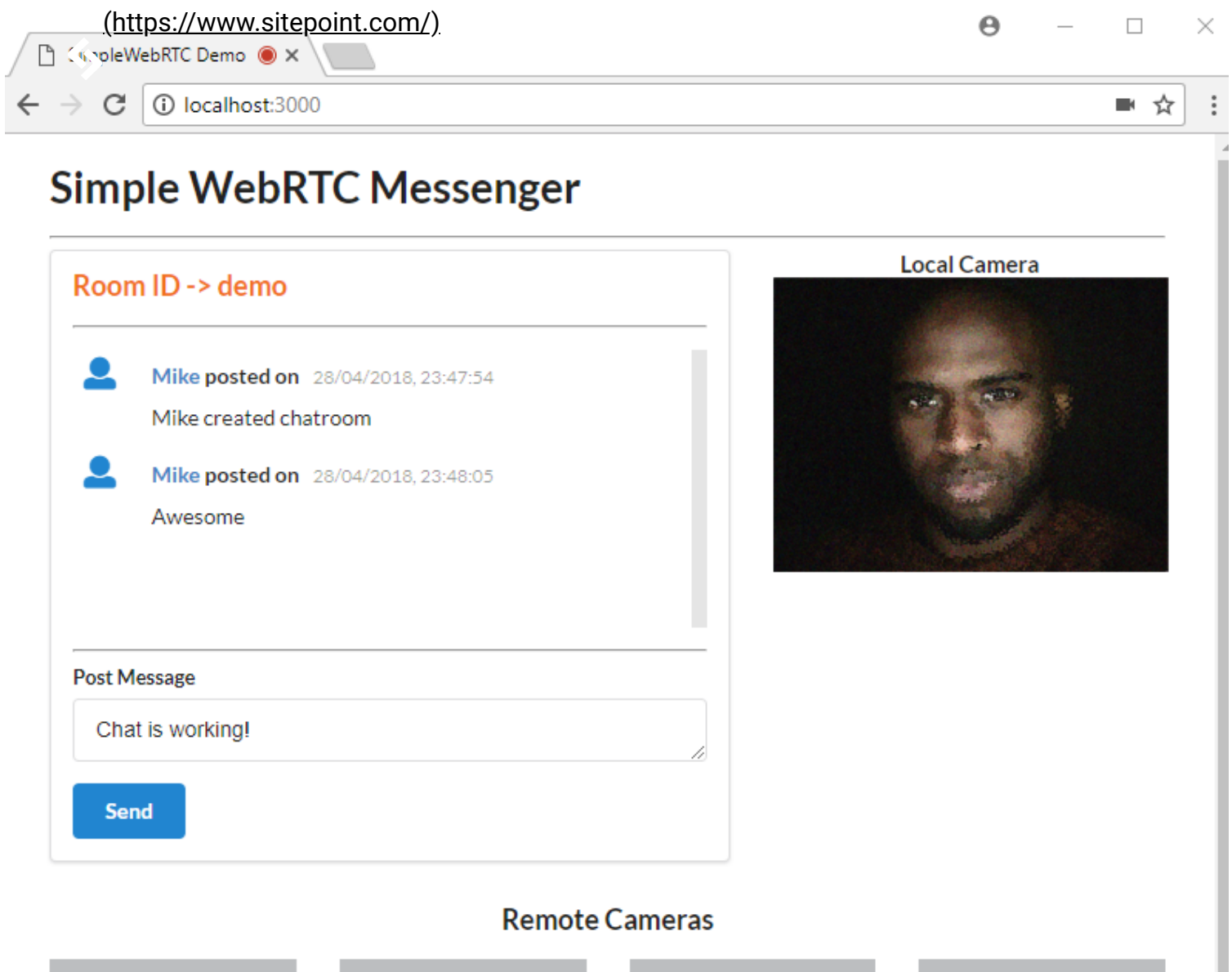
```
// Receive message from remote user  
webRTC.connection.on('message', (data) => {  
  if (data.type === 'chat') {  
    const message = data.payload;  
    messages.push(message);  
    updateChatMessages();  
  }  
});
```

That's all the logic we need to have the chat room work. Refresh the page and log in:



Hit the *Create Room* button. You'll be taken to this view. Post some messages to confirm the chat room is working.

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!



Once you've confirmed it's working, move on to the next task.

Remote Video Camera

As mentioned earlier, SimpleWebRTC supports multiple peers. Here's the code for adding remote video streams when a new user joins a room:

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy. \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!

```

    (https://www.sitepoint.com/).
    // Remote video was added
    webrtc.on('videoAdded', (video, peer) => {
      const id = webrtc.getDomId(peer);
      const html = remoteVideoTemplate({ id });
      if (remoteVideosCount === 0) {
        remoteVideosEl.html(html);
      } else {
        remoteVideosEl.append(html);
      }
      $(`#${id}`).html(video);
      $(`#${id} video`).addClass('ui image medium'); // Make video element responsive
      remoteVideosCount += 1;
    });

```

That's it. I'm sorry if you were expecting something more complicated. What we did is simply add an event listener for **videoAdded**, the callback of which receives a **video** element that can be directly add to the DOM. It also receives a **peer** object that contains useful information about our peer connection, but in this case, we're only interested in the DOM element's ID.

Unfortunately, testing this bit of code isn't possible without running it on an HTTPS server. Theoretically, you can generate a self-signed certificate for your Express server in order to run the app within your internal network. But the bad news is that browsers won't allow you to access the webcam if the certificate isn't from a trusted authority.

The simplest solution to testing the above code is to deploy it to a public server that supports the HTTPS protocol.

Deployment

This method that we're about to perform is one of the easiest ways to deploy a NodeJS app. All we have to do is first register an account with **now.sh** (<https://zeit.co/now>) we're happy to help!

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service](#) ([legals](#)), and [Privacy Policy](#). ([privacy-policy](#)). Have questions? Please contact support@sitepoint.com

Simply choose the free plan. You'll need to provide your email address. You'll also need to verify your email address for your account to activate. Next, install **now** CLI tool on your system:

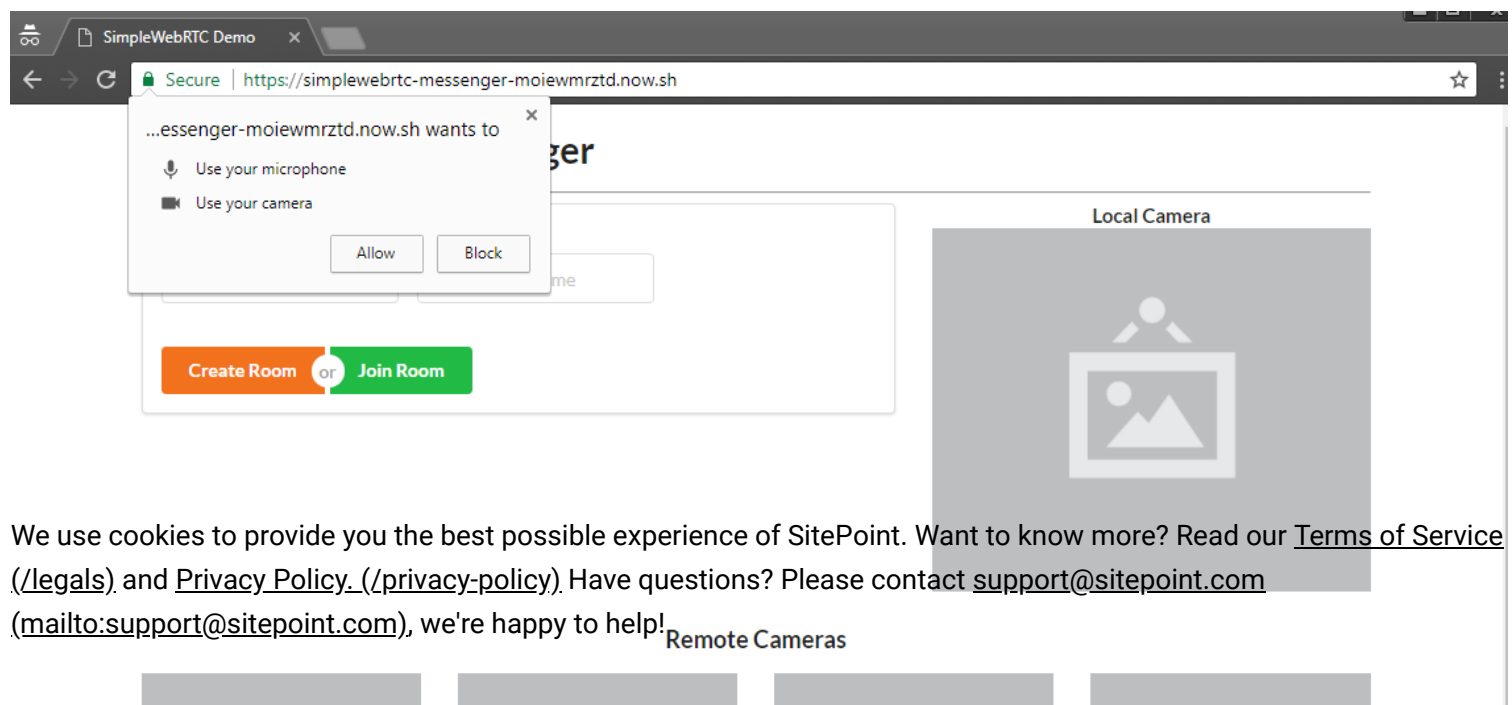
```
npm install -g now
```

After installation is complete, you can deploy the application. Simply execute the following command at the root of your project folder:

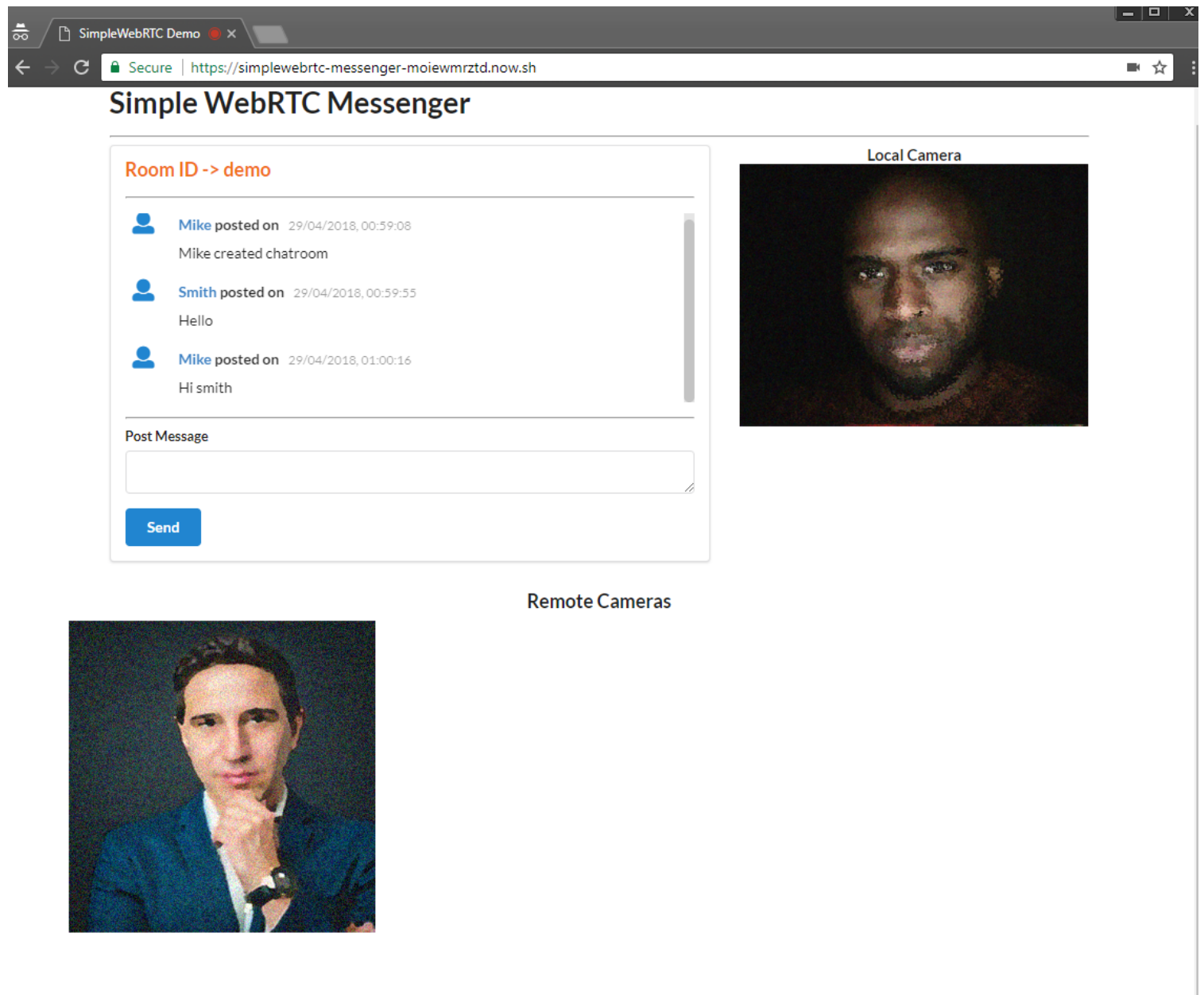
```
now --public
```

If this is the first time you're running the command, you'll be asked to enter your email address. You'll then receive an email that you'll need in order to verify your login. After verification has been done, you'll need to execute the command **now --public** again. After a few seconds, your app will be up and running at a specified URL that will be printed out on the terminal.

If you're using the VSCode integrated terminal, simply press **ALT** and click to open the URL in your browser.



(<https://www.sitepoint.com/>)
You'll need to allow the page to access your camera and microphone. Next create a room just like before. After you've signed in, you need to access another device — such as another laptop or smartphone with a front-facing camera. You could also ask a friend with an internet connection to help you with this. Simply access the same URL, and enter a new username and the same room name. The remote user will have to hit the *Join Room* button. Within a few seconds, both devices should be connected to the chat room. If a device doesn't have a camera, that's okay, as the chat functionality will still work.



We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>), we're happy to help!



Conclusion

[\(https://www.sitepoint.com/\)](https://www.sitepoint.com/)
In this tutorial, you've learned about SimpleWebRTC and how you can use it to create real-time apps. Specifically we've created a messaging application that allows the user to send text and make a video call to a remote peer. SimpleWebRTC is a really great cross-browser library for painlessly implementing WebRTC in web applications.

Don't forget that the code used in this tutorial is available on GitHub [. \(https://github.com/sitepoint-editors/simplewebrtc-messenger\)](https://github.com/sitepoint-editors/simplewebrtc-messenger). Clone it, make something cool, and have fun!



Meet the author

[\(htt](https://www.sitepoint.com/author/mwanyoike/) **Michael Wanyoike** <https://www.sitepoint.com/author/mwanyoike/>) 
[. \(https://twitter.com/myxsys\)](https://twitter.com/myxsys) **in**
[. \(https://www.linkedin.com/in/mikewanyoike/\)](https://www.linkedin.com/in/mikewanyoike/) 
[. \(https://github.com/brandiga\)](https://github.com/brandiga)

I write clean, readable and modular code. I love learning new technologies that bring efficiencies and increased productivity to my workflow.

Stuff We Do

- [Premium \(/premium/\)](/premium/)
- [Versioning \(/versioning/\)](/versioning/)
- [Forums \(/community/\)](/community/)
- [References \(/html-css/css/\)](/html-css/css/)

About

- [Our Story \(/about-us/\)](/about-us/)
- [Press Room \(/press/\)](/press/)

Contact

- [Contact Us \(/contact-us/\)](/contact-us/)
- [FAQ \(https://sitepoint.zendesk.com/hc/en-us\)](https://sitepoint.zendesk.com/hc/en-us)
- [Write for Us \(/write-for-us/\)](/write-for-us/)
- [Advertise \(/advertise/\)](/advertise/)

Legals

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals/terms-of-use/\)](/legals/terms-of-use/) and [Privacy Policy \(/legals/privacy-policy/\)](/legals/privacy-policy/). Have questions? Please contact support@sitepoint.com [. \(mailto:support@sitepoint.com\)](mailto:support@sitepoint.com), we're happy to help!

- [Privacy Policy \(/legals/#privacy\)](/legals/#privacy)

[\(https://www.sitepoint.com/\)](https://www.sitepoint.com/)

Connect



[_ \(https://www.facebook.com/sitepoint\)](https://www.facebook.com/sitepoint)



[_ \(http://twitter.com/sitepointdotcom\)](http://twitter.com/sitepointdotcom)



[\(/versioning/\)](#)



[_ \(https://www.sitepoint.com/feed/\)](https://www.sitepoint.com/feed/)



[_ \(https://plus.google.com/+sitepoint\)](https://plus.google.com/+sitepoint)

© 2000 – 2018 SitePoint Pty. Ltd.

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#), and [Privacy Policy. \(/privacy-policy\)](#). Have questions? Please contact [support@sitepoint.com \(mailto:support@sitepoint.com\)](mailto:support@sitepoint.com), we're happy to help!