**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

---

**BÁO CÁO QUÁ TRÌNH**

**NGUYỄN VIẾT DANH**

**ĐỖ HOÀNG PHƯƠNG**

**TÌM HIỂU CÔNG NGHỆ WebRTC**

***Thủ Đức, ngày 27 tháng 04 năm 2016***

Mục lục

[Chương 1 CÁCH CẤU HÌNH HTTPS TRÊN WAMP SERVER 3](#_Toc449680985)

[1.1 Bước 1: tải và cài đặt WAMP 3](#_Toc449680986)

[1.2 Bước 2: tải và cài đặt OpenSSL 3](#_Toc449680987)

[1.3 Bước 3: cấu hình WAMP để sử dụng HTTPS = HTTP + SSL 6](#_Toc449680988)

[1.4 Bước 4: uncomment một số dòng trong file httpd.conf 11](#_Toc449680989)

[1.5 Bước 5: điều chỉnh file php.ini 11](#_Toc449680990)

[1.6 Bước 6: điều chỉnh file httpd-ssl.conf 11](#_Toc449680991)

[1.7 Bước 7: kiểm tra syntax 12](#_Toc449680992)

[Chương 2 MÔ TẢ DEMO WebRTC 13](#_Toc449680993)

# CÁCh CẤU HÌNH HTTPS TRÊN WAMP SERVER

## Bước 1: tải và cài đặt WAMP

###### Chọn ổ đĩa C:\ làm nơi cài đặt WAMP

###### Vào <http://www.wampserver.com/en/> để tải bản WAMP phù hợp. Bản WAMP đang sử dụng hiện tại là WAMPSERVER (64 BITS & PHP 5.5) 2.5

## Bước 2: tải và cài đặt OpenSSL

###### Vào <https://slproweb.com/products/Win32OpenSSL.html> để tải OpenSSL

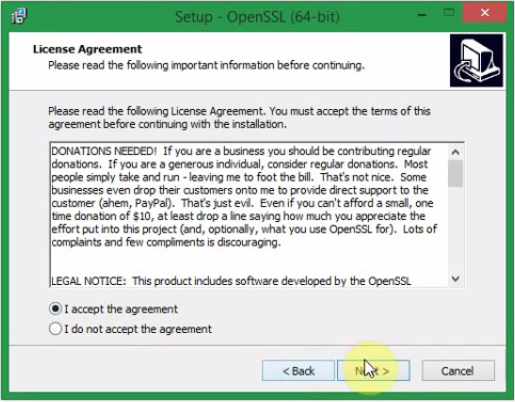
###### Chọn phiên bản *Win64 OpenSSL v1.0.2g*

###### Thực hiện cài OpenSSL như sau:

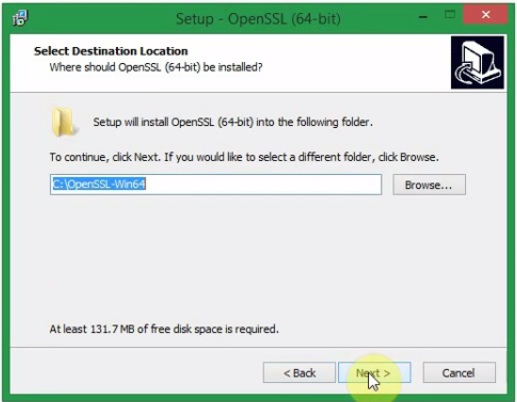
Mở bản cài đặt OpenSSL 🡪 Next



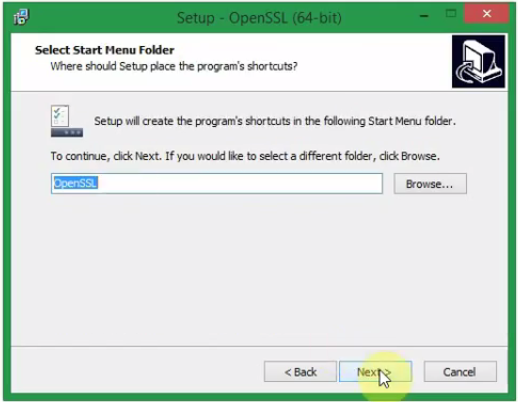
Chọn “I accept the agreement’ 🡪 Next



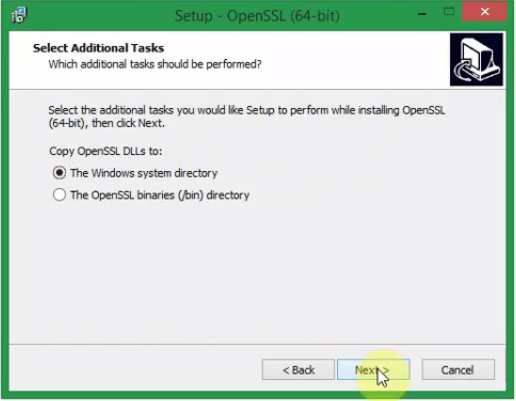
Chọn nơi cài đặt OpenSSL 🡪 Next

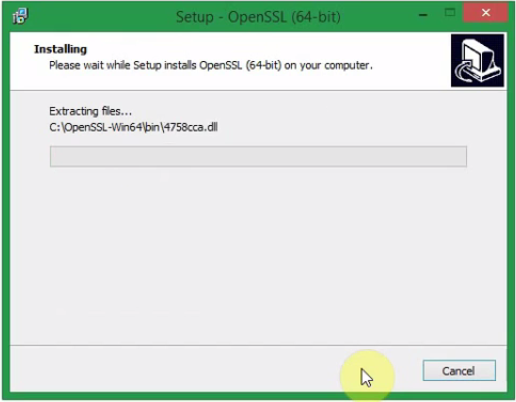


Chọn Next

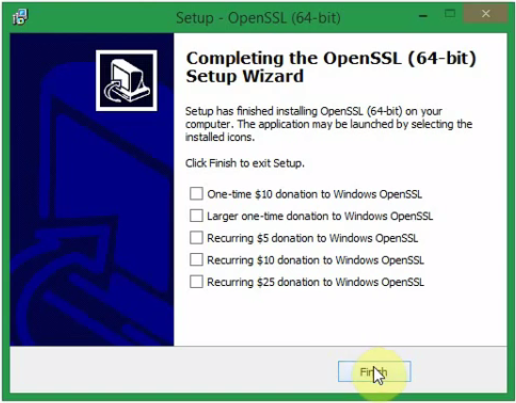


Chọn “The Windows system directory” 🡪 Next 🡪 Install



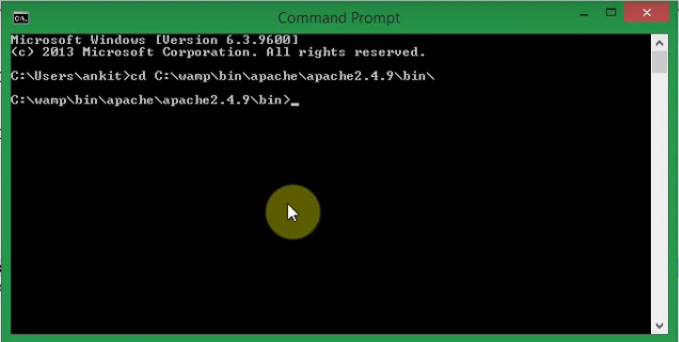


Cài đặt xong, bỏ chọn “One-time $10 donation to Windows OpenSSL” 🡪 Finish

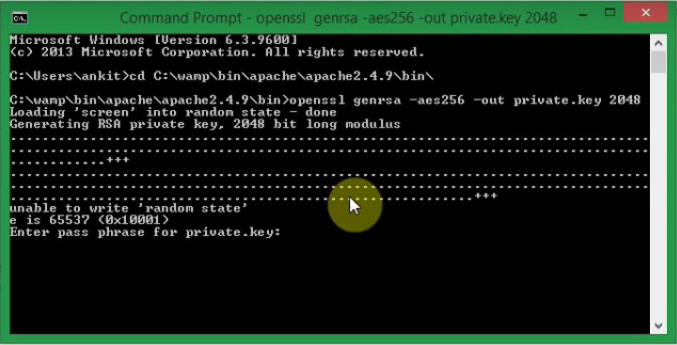


## Bước 3: cấu hình WAMP để sử dụng HTTPS = HTTP + SSL

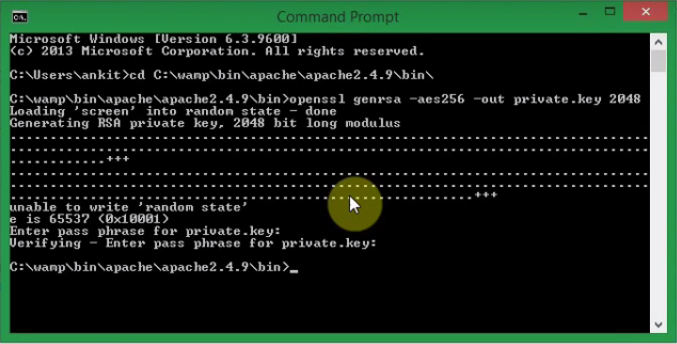
###### Mở Command Prompt và vào đường dẫn C:\wamp\bin\apache\apache2.4.9\bin\



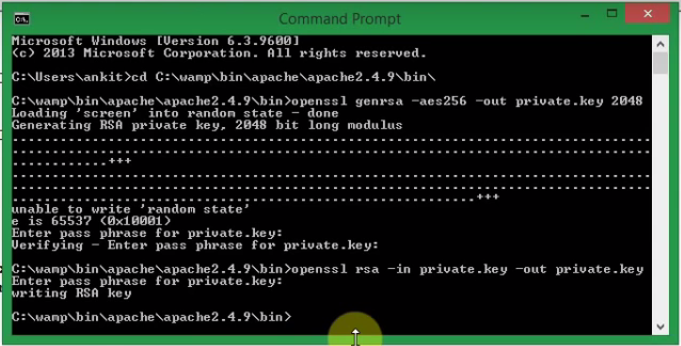
###### Thực hiện câu lệnh: *openssl genrsa -aes256 -out private.key 2048*



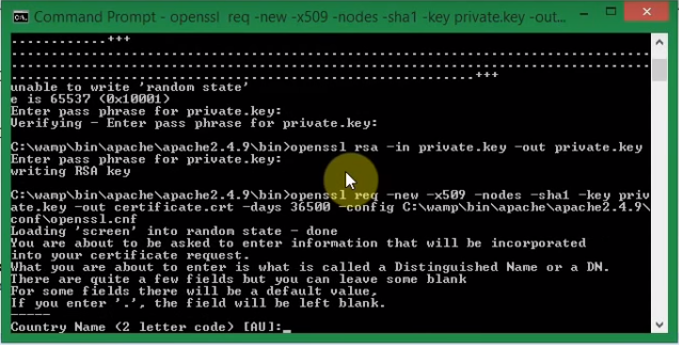
###### Điền private key là “*hellolingo*” và xác nhận



###### Thực hiện câu lệnh: *openssl rsa -in private.key -out private.key* và điền private key đã tạo trước đó.



###### Thực hiện câu lệnh: *openssl req -new -x509 -nodes -sha1 -key private.key -out certificate.crt -days 36500 -config D:\wamp\bin\apache\apache2.4.9\conf\openssl.cnf*



Điền **Country Name**: Vi

Điền **State or Province Name**: Ho Chi Minh

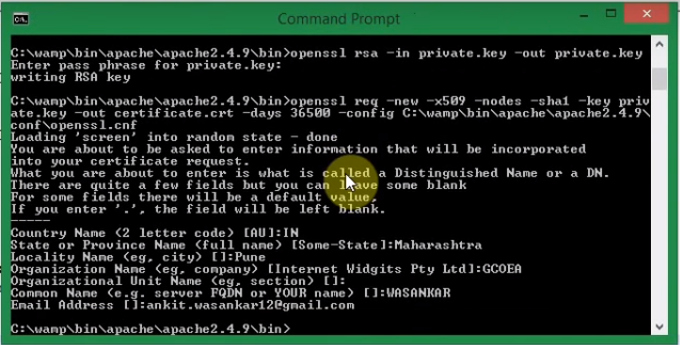
Điền **Locality Name**: Ho Chi Minh City

Điền **Organization Name**: SML

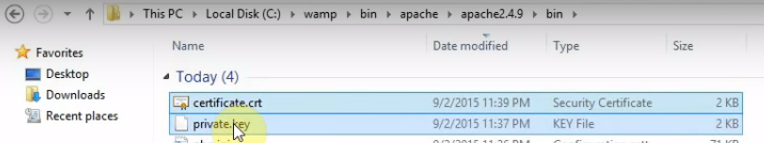
Điền **Organizational Unit Name**: Lab

Điền **Common Name**: Danh Nguyen

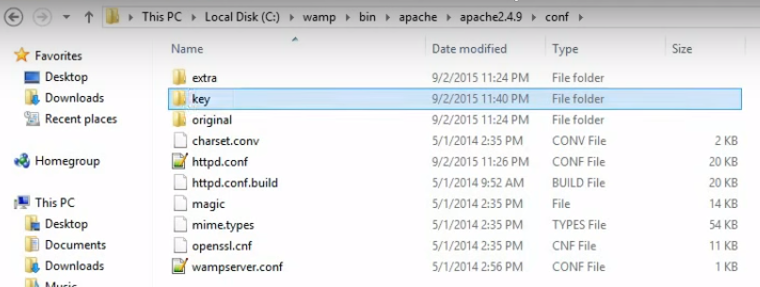
Điền **Email Address**: [danhnvuit@gmail.com](mailto:danhnvuit@gmail.com)



###### Vào thư mục *C:\wamp\bin\apache\apache2.4.9\bin*



###### Copy 2 tệp *certificate.crt* và *private.key* vào thư mục *C:\wamp\bin\apache\apache2.4.9\conf\* (nếu chưa có thư mục *key* thì tạo mới thư mục này)



## Bước 4: uncomment một số dòng trong file httpd.conf

###### Mở thư mục *C:\wamp\bin\apache\apache2.4.9\conf\* 🡪mở file httpd.conf

###### Uncomment 3 dòng sau:

*LoadModule ssl\_module modules/mod\_ssl.so*

*Include conf/extra/httpd-ssl.conf*

*LoadModule socache\_shmcb\_module modules/mod\_socache\_shmcb.so*

###### Lưu file (Ctrl + S)

## Bước 5: điều chỉnh file php.ini

###### Mở thư mục *C:\wamp\bin\php\php5.5.12\*

###### Mở file *php.ini*

###### Uncomment dòng *extension=php\_openssl.dll*

## Bước 6: điều chỉnh file httpd-ssl.conf

###### Mở thư mục *C:\wamp\bin\apache\apache2.4.9\conf\extra\*

###### Mở file *httpd-ssl.conf*

###### Tìm dòng *<VirtualHost \_default\_:443>* và điều chỉnh một số thông tin sau:

Thay đường dẫn của *DocumentRoot* là *C:/wamp/www*

Thay *ServerName* là *localhost:443*

Thay đường dẫn của *ErrorLog* là *C:/wamp/bin/apache/apache2.4.9/logs/ssl\_error.log*

Thay đường dẫn của *TransferLog* là *C:/wamp/bin/apache/apache2.4.9/logs/ssl\_access.log*

Thay đường dẫn của *CustomLog* là *C:/wamp/bin/apache/apache2.4.9/logs/ssl\_request.log*

Thay đường dẫn của *SSLCertificateFile* là *C:/wamp/bin/apache/apache2.4.9/conf/key/certificate.crt*

Thay đường dẫn của *SSLCertificateKeyFile* là *C:/wamp/bin/apache/apache2.4.9/conf/key/private.key*

###### Lưu file (Ctrl + S)

## Bước 7: kiểm tra syntax

###### Dùng lệnh *httpd –t* để kiểm tra syntax. Nếu Syntax OK là đã hoàn thành việc cấu hình HTTPS

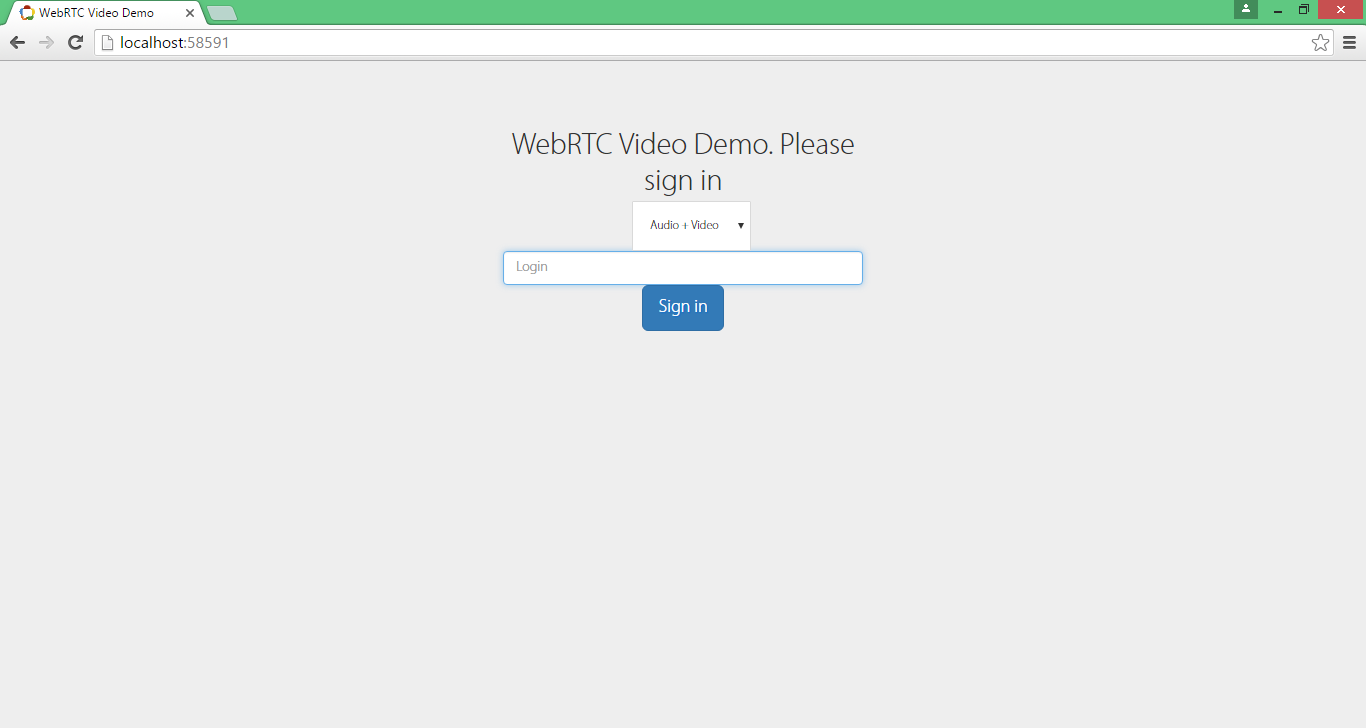
# MÔ TẢ DEMO WebRTC

Để tạo ra một kết nối WebRTC của user phải có khả năng truyền thông tin mà không cần sử dụng một kết nối ngang hàng WebRTC được.

* Bây giờ chúng ta cần tạo một trang HTML chính của dự án kết hợp với Bootstrap để tạo ra một chút chỉnh sửa riêng. Ta đặt tên *index.html* và đặt tập tin trong thư mục gốc với code sau đây:

|  |
| --- |
| <html>  <head>  <title>WebRTC Video Demo</title>  <link rel="stylesheet" href="node\_modules/bootstrap/dist/css/bootstrap.min.css" />  </head>  <body>  <div id="loginPage" class="container text-center">  <div class="row">  <div class="col-md-4 col-md-offset-4">  <h2>WebRTC Video Demo. Please sign in</h2>  <div>  <select id="broadcasting-option">  <option>Audio + Video</option>  </select>  <label for="usernameInput" class="sr-only">Login</label>  <input type="email" id="broadcast-name"  class="form-control formgroup" placeholder="Login"  required="" autofocus="">  </div>  <button id="setup-new-broadcast" class="btn btn-lg btn-primary btnblock">  Sign in  </button>  </div>  <!-- list of all available broadcasting rooms -->  <table style="width: 100%;" id="rooms-list"></table>  <!-- local/remote videos container -->  <div id="videos-container"></div>  </div>  </div>  </body>  </html> |

Chạy thử:



Đây là màn hình chính làm việc của dự án, gồm 3 mục chính:

* Combobox: chọn lựa kiểu (dự án sẽ phát triển lên chat, gửi màn hình…)
* Input: Nhập tên user
* Button: Đăng nhập vào hệ thống dự án.

Bây giờ chúng ta cần phải thiết lập kết nối với máy chủ truyền tín hiệu. Chúng ta sẽ sử dụng dịch vụ API là Firebase để lưu trữ và đồng bộ dữ liệu real-time. Có nghĩa là chúng ta không cần phải lo lắng về backend server, cơ sở dữ liệu, hay các thành phần real-time. Firebase sẽ xử lý tự động cho bạn.

* Ta sẽ viết những mã javascript để khởi tạo những kênh truyền và gọi nhau đầu tiên:

|  |
| --- |
| /\* UI specific \*/  var videosContainer = document.getElementById('videos-container') || document.body;  var setupNewBroadcast = document.getElementById('setup-new-broadcast');  var roomsList = document.getElementById('rooms-list');  var broadcastingOption = document.getElementById('broadcasting-option');  if (setupNewBroadcast)  setupNewBroadcast.onclick = setupNewBroadcastButtonClickHandler;  var config = {  openSocket: function (config) {  var channel = config.channel || location.href.replace(/\/|:|#|%|\.|\[|\]/g, '');  var socket = new Firebase('https://webrtc.firebaseIO.com/' + channel);  socket.channel = channel;  socket.on("child\_added", function (data) {  config.onmessage && config.onmessage(data.val());  });  socket.send = function (data) {  this.push(data);  };  config.onopen && setTimeout(config.onopen, 1);  socket.onDisconnect().remove();  return socket;  },  onRemoteStream: function (htmlElement) {  htmlElement.setAttribute('controls', true);  videosContainer.insertBefore(htmlElement, videosContainer.firstChild);  htmlElement.play();  rotateInCircle(htmlElement);  },  onRoomFound: function (room) {  var alreadyExist = document.querySelector('button[data-broadcaster="' + room.broadcaster + '"]');  if (alreadyExist) return;  if (typeof roomsList === 'undefined') roomsList = document.body;  var tr = document.createElement('tr');  tr.innerHTML = '<td><strong>' + room.roomName + '</strong> is broadcasting his media!</td>' + '<td><button class="join">Join</button></td>';  roomsList.insertBefore(tr, roomsList.firstChild);  var joinRoomButton = tr.querySelector('.join');  joinRoomButton.setAttribute('data-broadcaster', room.broadcaster);  joinRoomButton.setAttribute('data-roomToken', room.broadcaster);  joinRoomButton.onclick = function () {  this.disabled = true;  var broadcaster = this.getAttribute('data-broadcaster');  var roomToken = this.getAttribute('data-roomToken');  broadcastUI.joinRoom({  roomToken: roomToken,  joinUser: broadcaster  });  hideUnnecessaryStuff();  };  },  onNewParticipant: function (numberOfViewers) {  document.title = 'Viewers: ' + numberOfViewers;  }  }; |

Hãy chú ý vào biến “config” nhé: Nó được khởi tạo bởi một hàm sử dụng real-time nên sẽ được gọi lại theo thời gian để cập nhật cũng như thực hiện những nhiệm vụ mà bạn yêu cầu. Và gần như cả dự án của bạn từ đây phát sinh ra.

Bạn có thể nhìn thấy đoạn mã này: *location.href.replace( /\/|:|#|%|\.|\[|\]/g , '')*

Nó có nghĩa là "kênh" này là một định danh duy nhất của căn phòng được lấy từ các URL. Vì vậy, phòng của bạn sẽ chỉ hiển thị trên tên miền của bạn.

Bạn có thể sử dụng tên kênh tùy chỉnh nếu bạn muốn mở đơn kênh giữa tất cả các trang của bạn: ***var channel = config.channel || 'this-must-be-unique-must-be-private';***

Khi một kênh được tạo ra thì socket.on () bắt đầu lắng nghe sự kiện máy chủ gửi từ sails với các quy định “child\_added”. sẽ kích hoạt chức năng gọi lại cung cấp khi một sự kiện phù hợp là nhận được.

Hàm onRoomFound: function (room): Giữ nhiệm vụ tìm kiếm những user đã truy cập vào kênh và hiển thị ra màn hình.

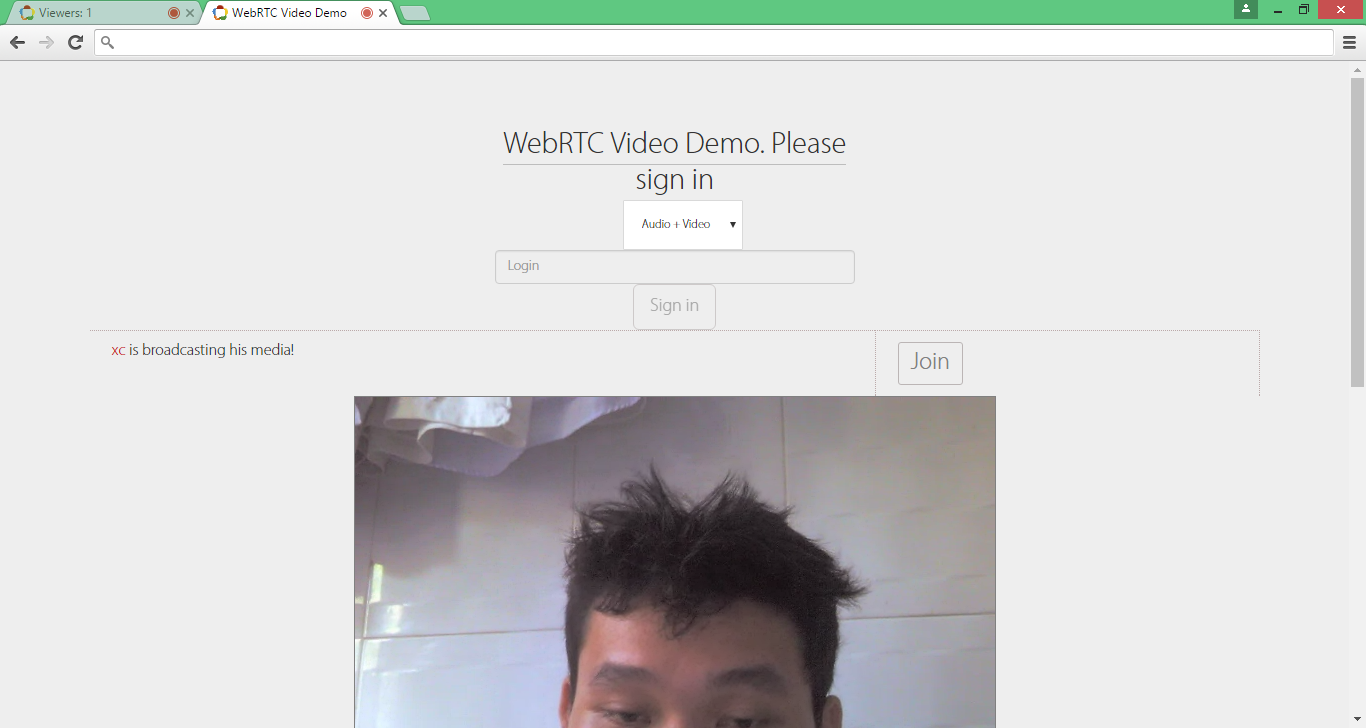
Với đoạn code ở cuối:

onNewParticipant: function (numberOfViewers) {

document.title = 'Viewers: ' + numberOfViewers;

}

Sẽ cho chúng ta biết hiện tại có bao nhiêu người kết nối và truy cập tới bạn.



* Thêm mã xử lý khi người dùng Login vào (Khi nhấn “Sign in”).

|  |
| --- |
| var broadcastUI = broadcast(config);  function setupNewBroadcastButtonClickHandler() {  document.getElementById('broadcast-name').disabled = true;  document.getElementById('setup-new-broadcast').disabled = true;  captureUserMedia(function () {  var shared = 'video';  if (window.option == 'Only Audio') {  shared = 'audio';  }  if (window.option == 'Screen') {  shared = 'screen';  }  broadcastUI.createRoom({  roomName: (document.getElementById('broadcast-name') || {}).value || 'Anonymous',  isAudio: shared === 'audio'  });  });  hideUnnecessaryStuff();  } |

var broadcastUI = broadcast(config);

Sẽ gọi qua file broadcast.js thực hiện khởi tạo một phòng nhằm phục vụ cho việc người dùng truy cập vào.

Với đoạn code này:

var shared = 'video';

if (window.option == 'Only Audio') {

shared = 'audio';

}

if (window.option == 'Screen') {

shared = 'screen';

}

Ta sẽ gọi tới hàm captureUserMedia với dữ liệu đầu vào là combobox ban đầu lựa chọn.

*Lưu ý: Screen không có do ban đầu phát triển việc chỉ gửi màn hình nhưng chưa làm xong.*

Với đoạn code sau:

broadcastUI.createRoom({

roomName: (document.getElementById('broadcast-name') || {}).value || 'Anonymous',

isAudio: shared === 'audio'

});

Tạo ra một phòng mới (chúng ta hay gọi là người dùng mới) với tên đăng nhập ban đầu trong input sẽ sử dụng để gọi nhau sau này nếu chúng ta không nhập gì sẽ mặc định là ‘Anonymous’.

* Mở kết nối với MicroPhone và Webcam

|  |
| --- |
| function captureUserMedia(callback) {  var constraints = null;  window.option = broadcastingOption ? broadcastingOption.value : '';  if (option === 'Only Audio') {  constraints = {  audio: true,  video: false  };  if (DetectRTC.hasMicrophone !== true) {  alert('DetectRTC library is unable to find microphone; maybe you denied microphone access once and it is still denied or maybe microphone device is not attached to your system or another app is using same microphone.');  }  }  if (option === 'Screen') {  var video\_constraints = {  mandatory: {  chromeMediaSource: 'screen'  },  optional: []  };  constraints = {  audio: false,  video: video\_constraints  };  if (DetectRTC.isScreenCapturingSupported !== true) {  alert('DetectRTC library is unable to find screen capturing support. You MUST run chrome with command line flag "chrome --enable-usermedia-screen-capturing"');  }  }  if (option != 'Only Audio' && option != 'Screen' && DetectRTC.hasWebcam !== true) {  alert('DetectRTC library is unable to find webcam; maybe you denied webcam access once and it is still denied or maybe webcam device is not attached to your system or another app is using same webcam.');  }  var htmlElement = document.createElement(option === 'Only Audio' ? 'audio' : 'video');  htmlElement.setAttribute('autoplay', true);  htmlElement.setAttribute('controls', true);  videosContainer.insertBefore(htmlElement, videosContainer.firstChild);  var mediaConfig = {  video: htmlElement,  onsuccess: function (stream) {  config.attachStream = stream;  callback && callback();  htmlElement.setAttribute('muted', true);  rotateInCircle(htmlElement);  },  onerror: function () {  if (option === 'Only Audio') alert('unable to get access to your microphone');  else if (option === 'Screen') {  if (location.protocol === 'http:') alert('Please test this WebRTC experiment on HTTPS.');  else alert('Screen capturing is either denied or not supported. Are you enabled flag: "Enable screen capture support in getUserMedia"?');  } else alert('unable to get access to your webcam');  }  };  if (constraints) mediaConfig.constraints = constraints;  getUserMedia(mediaConfig);  } |

* Tiếp theo ta thêm phần tạo hiệu ứng quay khung video 360

|  |
| --- |
| function rotateInCircle(video) {  video.style[navigator.mozGetUserMedia ? 'transform' : '-webkit-transform'] = 'rotate(0deg)';  setTimeout(function () {  video.style[navigator.mozGetUserMedia ? 'transform' : '-webkit-transform'] = 'rotate(360deg)';  }, 1000);  } |

* **Và dăc biệt ta cần thêm một số thư viện thực hiện:**

|  |
| --- |
| <link rel="stylesheet" href="cdn.webrtc-experiment.com/style.css">  <!-- scripts used for broadcasting -->  <script src="cdn.webrtc-experiment.com/firebase.js"> </script> <!-- Signaling -->  <script src="cdn.webrtc-experiment.com/RTCPeerConnection-v1.5.js"> </script> <!-- WebRTC simple wrapper -->  <script src="cdn.webrtc-experiment.com/webrtc-broadcasting/broadcast.js"> </script> <!-- Multi-user connectivity handler -->  <!-- This Library is used to detect WebRTC features -->  <script src="cdn.webrtc-experiment.com/DetectRTC.js"></script>  <style>  body {  background: #eee;  padding: 5% 0;  }  video {  background: black;  border: 1px solid gray;  }  .call-page {  position: relative;  display: block;  margin: 0 auto;  width: 500px;  height: 500px;  }  #localVideo {  width: 150px;  height: 150px;  position: absolute;  top: 15px;  right: 15px;  }  #remoteVideo {  width: 500px;  height: 500px;  }  </style> |

firebase.js : Các phần truyền tín hiệu, làm việc với socket.

RTCPeerConnection-v1.5.js: Phần quan trọng

* RTCPeerConnection: Được xem là phần quan trọng nhất của WebRTC. Cung cấp tất cả khả năng giao tiếp của WebRTC. Nó trao đổi khả năng sử dụng SDP, chạy thuật toán vượt NAT bằng việc gửi và nhận các phương tiện của chính nó.
* getUserMedia:Có thể kết nối với Webcam và Microphone.
* createCORSRequest: Dùng để truy cập vào dịch vụ web và gửi dữ liệu

broadcast.js: Kết nối và xử lý đa người dùng. Tạo ra một broadcast như một phòng và thực hiện nhiệm vụ cho 2 hay nhiều người kết nối với nhau.

DetectRTC.js: Thực hiện các chức năng của WebRTC. Thực hiện kiểm tra công cụ truy cập các thiết bị sử dụng địa chỉ IP…

* Chạy thử và xem kết quả:

Có 2 user truy cập vào phòng và bạn có thể click vào Join để liên lạc tương ứng với user đó.

