# Chat room

In this project, you will create a simple chat room application. Anyone that is running your application will be able to send messages to other users over the Internet.

## Core concepts

* Communicating and storing things over the Internet.
* Integrating software that is written by a third party.

So far, you have been building sofware that runs on your personal machine. You have been accessing data that exists on your machine or has been fetched for you by your browser.

In this project, we will show you how to communicate with other machines and store data on the Internet.

To do this, you need to use a server. A server is a computer that is always online and connected to the Internet.

Setting up and running your own server can be expensive and difficult. Firebase is a company that has servers, and allows you to build applications that use them. The chat application that you will build will use Firebase's servers to allow users to communicate with each other.

For more information on how Firebase works, visit www.firebase.com and read through the documentation.

## Files

As usual, create a new folder for this project and add the following files: manifest.json, chat.html, chat.css, chat.js

*manifest.json*

The only unique part of this manifest is the "content\_security\_policy" section. Because your application will be accessing resources on the Internet, we need to instruct Chrome to allow certain URLs to be requested. Normally, Chrome applications are not allowed to make requests to remote resources on the Internet.

{  
 "name": "My Chat Room",  
 "description": "This app lets you chat with other people.",  
 "version": "1.0",  
 "manifest\_version": 2,  
  
 "content\_security\_policy":  
 "script-src 'self' https://\*.firebase.com https://\*.firebaseio.com; object-src 'self'",  
  
 "app": {  
 "launch": {  
 "local\_path": "chat.html"  
 }  
 }  
}

*chat.html*

<!DOCTYPE html>  
<html>  
<head>  
 <!--  
 This next line will load some JavaScript from Firebase's servers. Since our  
 Javascript depends on Firebase, we must load it before our own code.  
 -->  
 <script type='text/javascript' src='https://cdn.firebase.com/v0/firebase.js'></script>  
  
 <!--  
 Now that Firebase has been loaded, we can load our JavaScript  
 -->  
 <script src="chat.js" type="text/javascript"></script>  
  
 <link rel="stylesheet" type="text/css" href="chat.css">  
</head>  
<body>  
 <h1>Chat room</h1>  
  
 <ul id='chat-message-list'>  
 </ul>  
  
 <form id='chat-form'>  
 <input type="text" name="message" id="message-input">  
 </form>  
</body>  
</html>

*chat.css*

The parts of the code between /\* and \*/ are "comments". You do not need to type in the comments.

#chat-message-list {  
 /\* Give the message list a height of 500 pixels \*/  
 height: 500px;  
  
 /\* The list will have a solid 1 pixel border. \*/  
 border: 1px solid grey;  
  
 /\* This tells the browser that the message list should add a scrollbar to  
 the y-axis \*/  
 overflow-y: auto;  
  
 /\* Add 15 pixels of blank space between the border of the message list and its  
 contents. \*/  
 padding: 15px;  
}  
  
#chat-message-list li {  
 /\* The browser usually gives lists a default styling, but we want to turn  
 that off. \*/  
 list-style-type: none;  
 padding: 0;  
}  
  
input[type=text] {  
 // The text input should be made as wide as possible.  
 width: 100%;  
}

*chat.js*

The parts of the code between /\* and \*/ are called "comments". They serve no functional purpose, other than to inform the reader. You do not need to type in the comments.

/\*  
 \* We connect to the remote Firebase server and store the connection into the  
 \* `chatRoom` variable.  
 \*  
 \* You may want to change the URL below to be something unique. For example,  
 \* you might change it to:  
 \*  
 \* 'https://dcode.firebaseio.com/chat\_room\_for\_dog\_owners'  
 \* or  
 \* 'https://dcode.firebaseio.com/chat\_room\_football\_fans'  
 \*  
 \* Every unique URL will create a new chatroom.  
 \*/  
var chatRoom = new Firebase('https://dcode.firebaseio.com/chat\_room');  
  
/\*  
 \* This tells the browser to wait until the DOMContentLoaded event occurs before  
 \* continuing. This is a common pattern found in Javascript applications.  
 \* Google for `DOMContentLoaded` for more information.  
 \*/  
document.addEventListener('DOMContentLoaded', function() {  
  
 /\*  
 \* We use the document.getElementById function to return references to  
 \* elements in the chat.html file. We store these references in variables  
 \* and are then able to query and manipulate their state.  
 \*/  
 var messageList = document.getElementById('chat-message-list');  
 var form = document.getElementById('chat-form');  
 var messageInput = document.getElementById('message-input');  
  
 /\*  
 \* We tell firebase to notify our application when a new chat message is  
 \* added. This message may be coming from a remote user, or it may be from the  
 \* user that is running the application locally.  
 \*/  
 chatRoom.on('child\_added', function(snapshot) {  
  
 /\*  
 \* `document.createElement` is a function that allows us to add new elements  
 \* to the HTML file. In this case, we are creating an `LI` element, inserting  
 \* the text of the new message, and adding it to the list.  
 \*/  
 var li = document.createElement('li');  
 li.textContent = snapshot.val().message;  
 messageList.appendChild(li);  
  
 /\*  
 \* This forces the message list to scroll to the bottom. If we didn't do  
 \* this, the user would have to scroll to the bottom of the list manually  
 \* whenever a new message was received.  
 \*/  
 messageList.scrollTop = messageList.scrollHeight;  
 });  
  
 /\*  
 \* We are telling the browser to run this code whenever a user enters a  
 \* new message into the form field  
 \*/  
 form.addEventListener('submit', function(event) {  
 /\*  
 \* Usually when a form is submitted, the browser will reload the page.  
 \* We don't want this to happen, so we call the event.preventDefault()  
 \* function  
 \*/  
 event.preventDefault();  
  
 /\*  
 \* We tell Firebase to add a new message to the chatroom. This will  
 \* cause the `child\_added` event to fire for every application that is  
 \* listening this chat room  
 \*/  
 chatRoom.push({message: messageInput.value});  
  
 /\*  
 \* Now that the message has been sent, we can clear out the input form.  
 \* If we didn't do this, the user would have to delete the text in the form  
 \* manually.  
 \*/  
 messageInput.value = '';  
 });  
});

## Running the application

Follow the same proceedure in the "hello world" project to install your chat room app into Chrome.

Once the app is installed, open a new tab tab. There should be an icon called "Chat Room".

## Improvement ideas

This is a very basic chat room. There is a lot of room for improvement. For example:

* Let users to identify themselves with a nickname. Send the nickname with each message.
* Let users to upload pictures into the chatroom.
* Let users join multiple chat rooms.

If you want to see what else you can with Firebase, go look at some of their demos online. All the demos include HTML and Javascript source code.

https://www.firebase.com/docs/examples.html

By integrating with Firebase, your application gained functionality that would have been difficult to build yourself. Integrating with third party services is a great way to quickly add rich features to your application.

In addition to Firebase, there are many more companies that provide services for developers. For example: Facebook, Twitter, Google, Instram, and many more. Try Googling around to see what you can find.