ASP.NET Core 2.1

Version

2.2 Preview 2

2.1

2.0

1.1

1.0

Tutorial: Get started with SignalR on ASP.NET Core

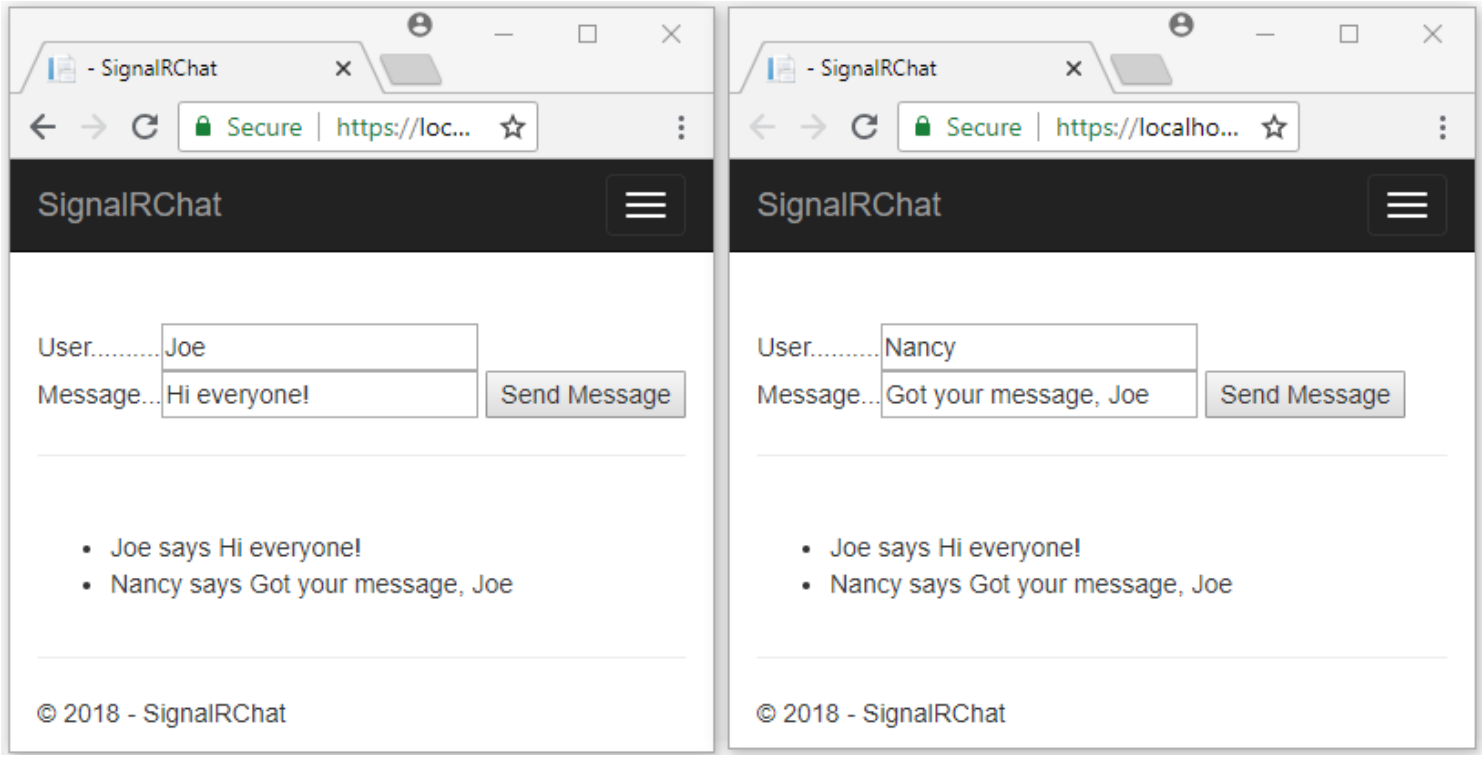
08/31/2018 6 minutes to read Contributors all

In this article

This tutorial teaches the basics of building a real-time app using SignalR. You learn how to:

- ✓ Create a web app that uses SignalR on ASP.NET Core.
- ✓ Create a SignalR hub on the server.
- ✓ Connect to the SignalR hub from JavaScript clients.
- ✓ Use the hub to send messages from any client to all connected clients.

At the end, you'll have a working chat app:



View or download sample code (how to download).

Prerequisites

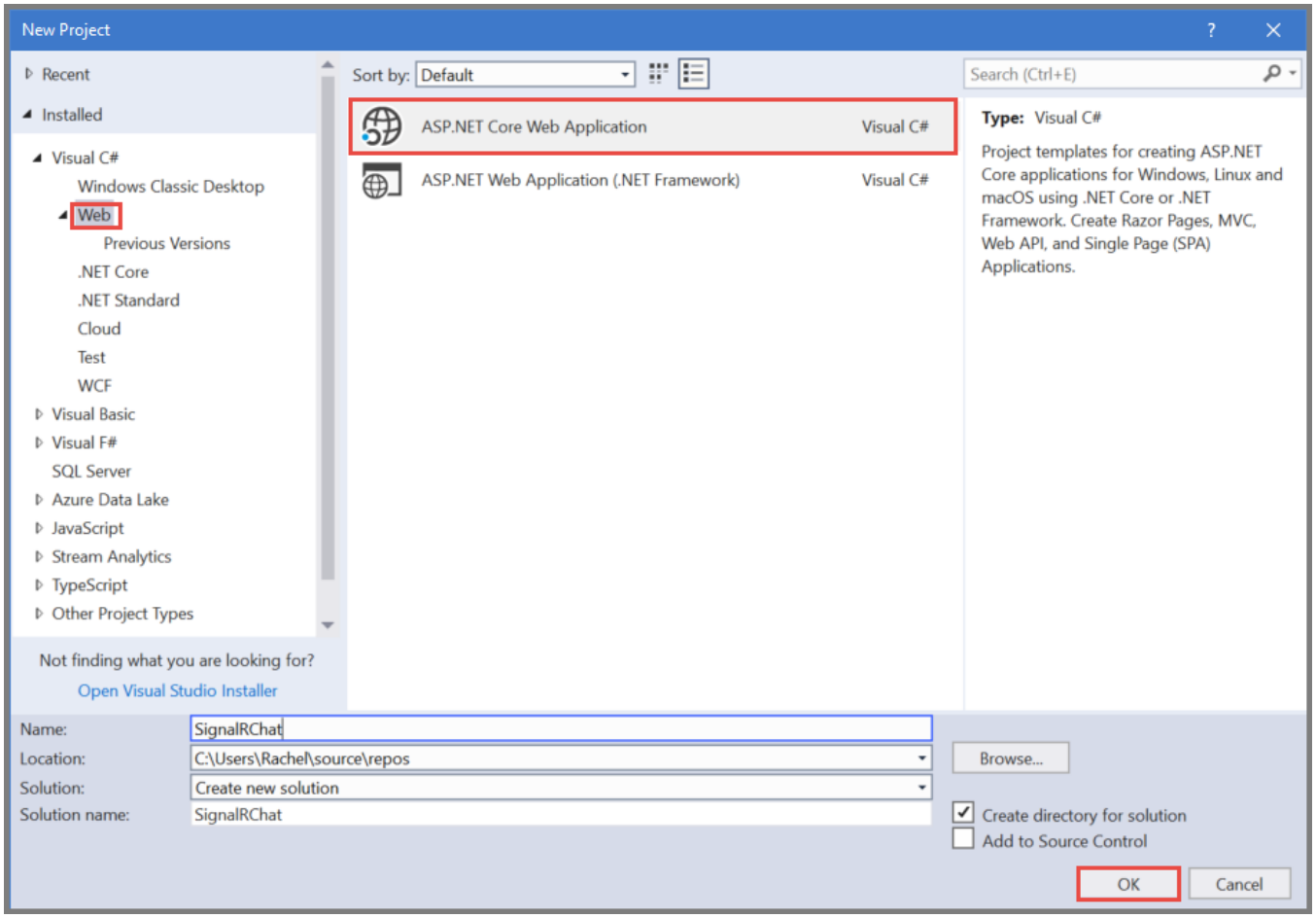
Visual Studio | Visual Studio Code | Visual Studio for Mac

- Visual Studio 2017 version 15.8 or later with the **ASP.NET and web development** workload
- .NET Core SDK 2.1 or later

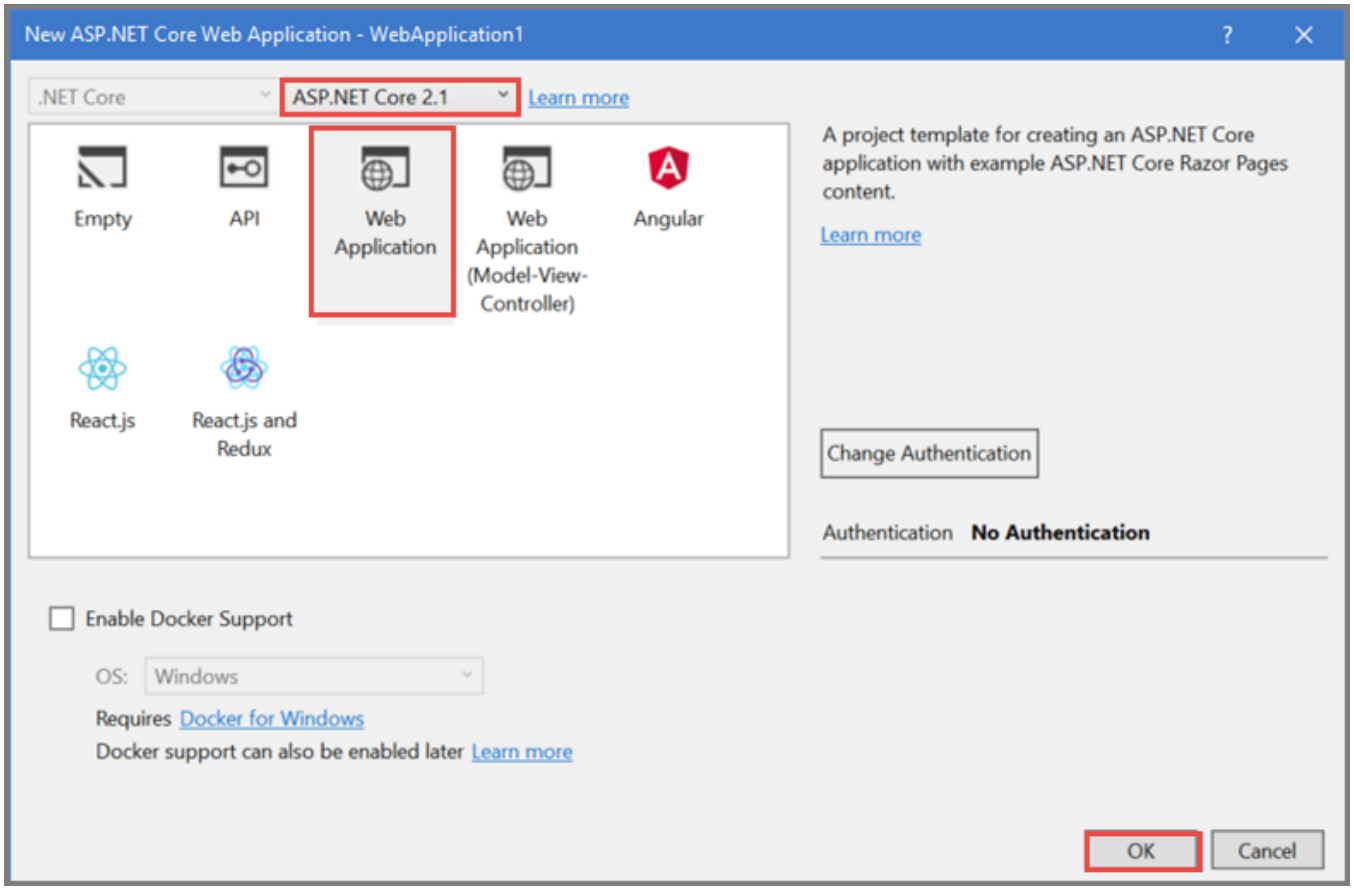
Create the project

Visual Studio | Visual Studio Code | Visual Studio for Mac

- From the menu, select **File** > **New Project**.
- In the **New Project** dialog, select **Installed** > **Visual C#** > **Web** > **ASP.NET Core Web Application**. Name the project *SignalRChat*.



- Select **Web Application** to create a project that uses Razor Pages.
- Select a target framework of **.NET Core**, select **ASP.NET Core 2.1**, and click **OK**.

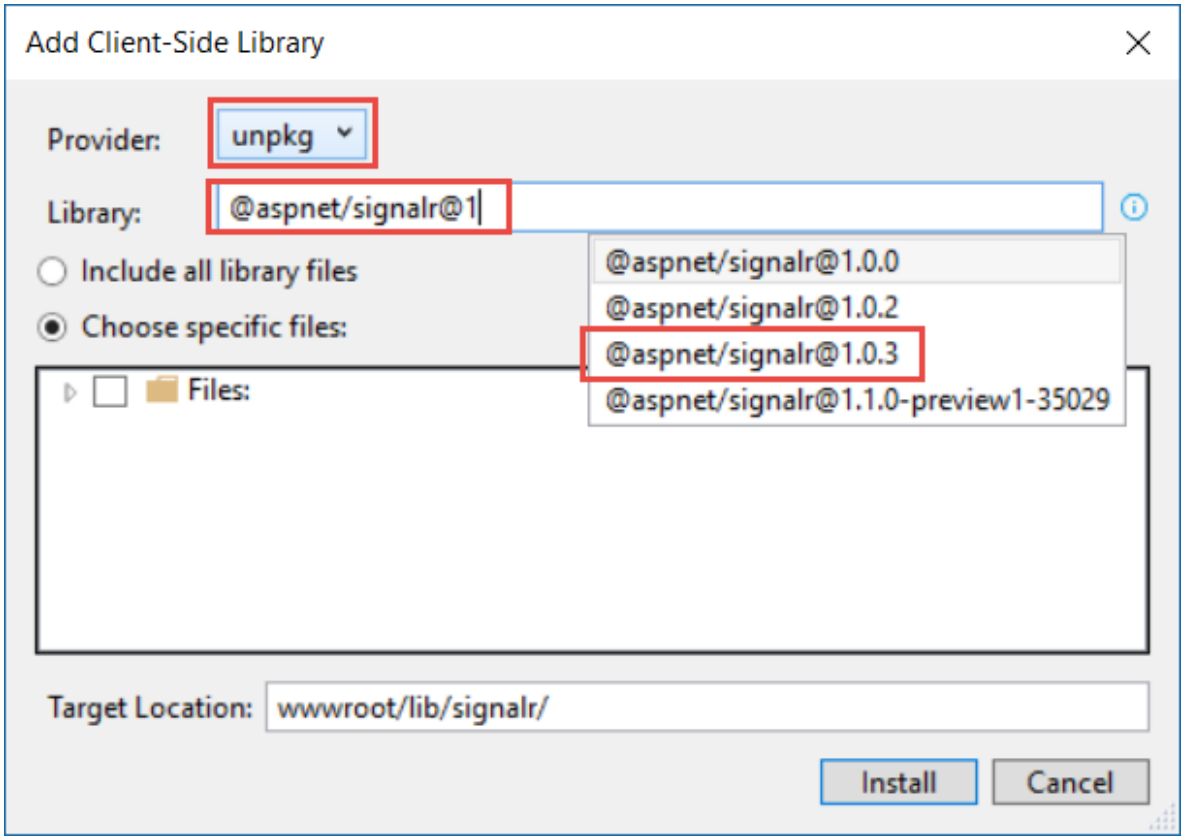


Add the SignalR client library

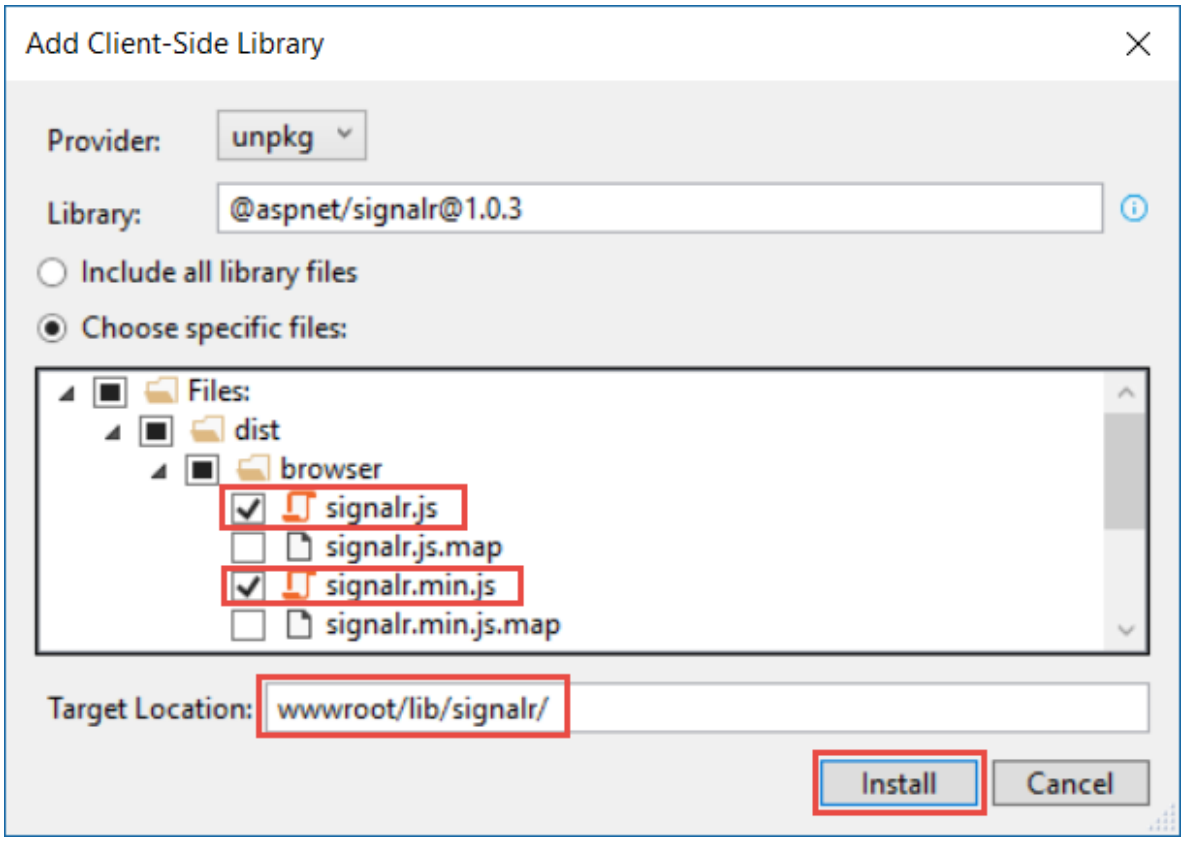
The SignalR server library is included in the [Microsoft.AspNetCore.App metapackage](). The JavaScript client library isn't automatically included in the project. For this tutorial, you use [Library Manager (LibMan)]() to get the client library from *unpkg*. [unpkg]() is a [content delivery network]() that can deliver anything found in [npm, the Node.js package manager]().

Visual Studio | Visual Studio Code | Visual Studio for Mac

- In **Solution Explorer**, right-click the project, and select **Add** > **Client-Side Library**.
- In the **Add Client-Side Library** dialog, for **Provider** select **unpkg**.
- For **Library**, enter _@aspnet/signalr@1_, and select the latest version that isn't preview.



- Select **Choose specific files**, expand the *dist/browser* folder, and select *signalr.js* and *signalr.min.js*.
- Set **Target Location** to *wwwroot/lib/signalr/*, and select **Install**.



LibMan creates a *wwwroot/lib/signalr* folder and copies the selected files to it.

Create the SignalR hub

A hub is a class that serves as a high-level pipeline that handles client-server communication.

- In the SignalRChat project folder, create a *Hubs* folder.
- In the *Hubs* folder, create a *ChatHub.cs* file with the following code:

C# Copy

```
using Microsoft.AspNetCore.SignalR;
using System.Threading.Tasks;

namespace SignalRChat.Hubs
{
    public class ChatHub : Hub
    {
        public async Task SendMessage(string user, string message)
        {
            await Clients.All.SendAsync("ReceiveMessage", user, message);
        }
    }
}
```

The `ChatHub` class inherits from the SignalR Hub class. The `Hub` class manages connections, groups, and messaging.

The `SendMessage` method can be called by any connected client. It sends the received message to all clients. SignalR code is asynchronous to provide maximum scalability.

Configure the project to use SignalR

The SignalR server must be configured to pass SignalR requests to SignalR.

- Add the following highlighted code to the *Startup.cs* file.

C# Copy

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
```

```
using Microsoft.Extensions.DependencyInjection;
using SignalRChat.Hubs;

namespace SignalRChat
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.Configure<CookiePolicyOptions>(options =>
            {
                // This lambda determines whether user consent for non-essential cookies
is needed for a given request.
                options.CheckConsentNeeded = context => true;
                options.MinimumSameSitePolicy = SameSiteMode.None;
            });


            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);

            services.AddSignalR();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Error");
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseCookiePolicy();
            app.UseSignalR(routes =>
            {
                routes.MapHub<ChatHub>("/chatHub");
            });
```

```
            app.UseMvc();
        }
    }
}
```

These changes add SignalR to the dependency injection system and the middleware pipeline.

Create the SignalR client code

- Replace the content in *Pages\Index.cshtml* with the following code:

CSHTML Copy

```
@page
<div class="container">
    <div class="row"> </div>
    <div class="row">
        <div class="col-6"> </div>
        <div class="col-6">
            User.....<input type="text" id="userInput" />
            <br />
            Message...<input type="text" id="messageInput" />
            <input type="button" id="sendButton" value="Send Message" />
        </div>
    </div>
    <div class="row">
        <div class="col-12">
            <hr />
        </div>
    </div>
    <div class="row">
        <div class="col-6"> </div>
        <div class="col-6">
            <ul id="messagesList"></ul>
        </div>
    </div>
</div>
<script src="~/lib/signalr/dist/browser/signalr.js"></script>
<script src="~/js/chat.js"></script>
```

The preceding code:

- Creates text boxes for name and message text, and a submit button.
- Creates a list with `id="messagesList"` for displaying messages that are received from the SignalR hub.
- Includes script references to SignalR and the *chat.js* application code that you create in the next step.

- In the *wwwroot/js* folder, create a *chat.js* file with the following code:

JavaScript Copy

```
"use strict";

var connection = new signalR.HubConnectionBuilder().withUrl("/chatHub").build();

connection.on("ReceiveMessage", function (user, message) {
    var msg = message.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g, "&gt;");
    var encodedMsg = user + " says " + msg;
    var li = document.createElement("li");
    li.textContent = encodedMsg;
    document.getElementById("messagesList").appendChild(li);
});

connection.start().catch(function (err) {
    return console.error(err.toString());
});

document.getElementById("sendButton").addEventListener("click", function (event) {
    var user = document.getElementById("userInput").value;
    var message = document.getElementById("messageInput").value;
    connection.invoke("SendMessage", user, message).catch(function (err) {
        return console.error(err.toString());
    });
    event.preventDefault();
});
```

The preceding code:

- Creates and starts a connection.
- Adds to the submit button a handler that sends messages to the hub.
- Adds to the connection object a handler that receives messages from the hub and adds them to the list.

Run the app

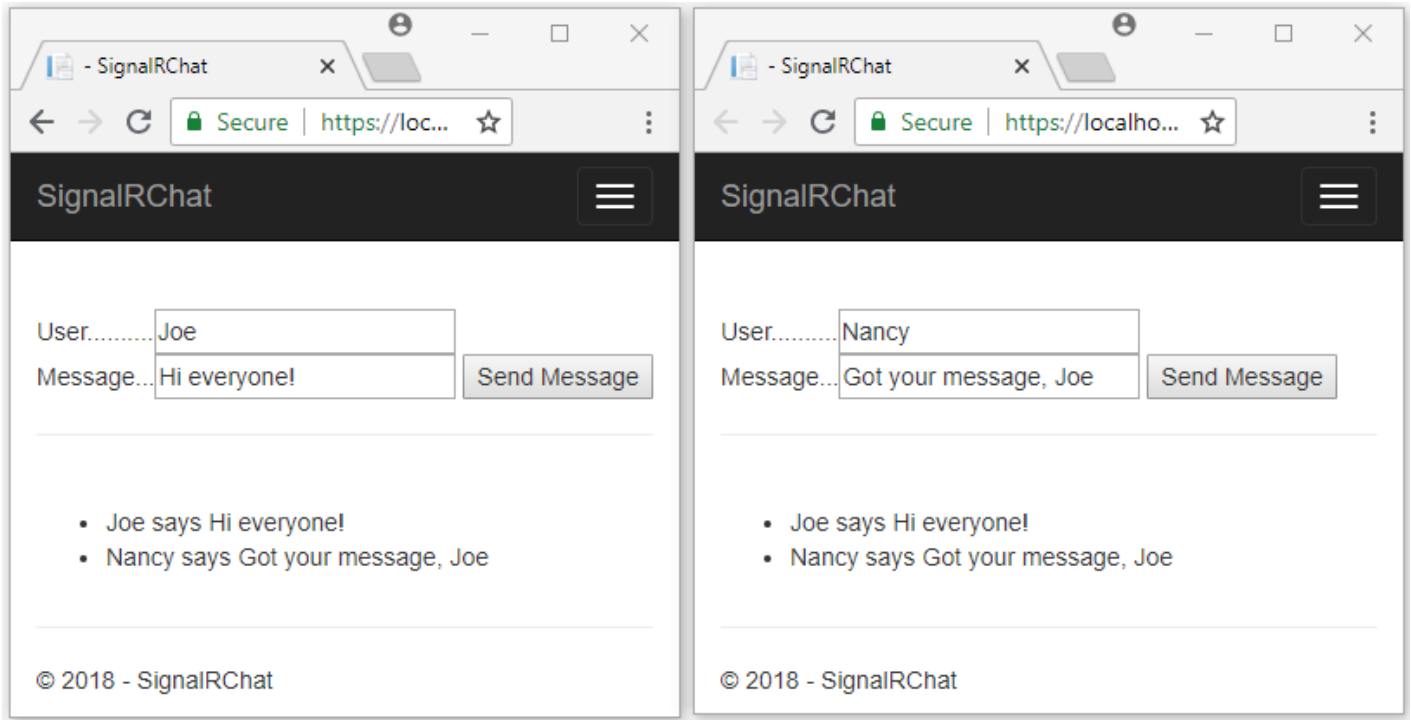
Visual Studio | Visual Studio Code | Visual Studio for Mac

- Press **CTRL+F5** to run the app without debugging.

- Copy the URL from the address bar, open another browser instance or tab, and paste the URL in the address bar.

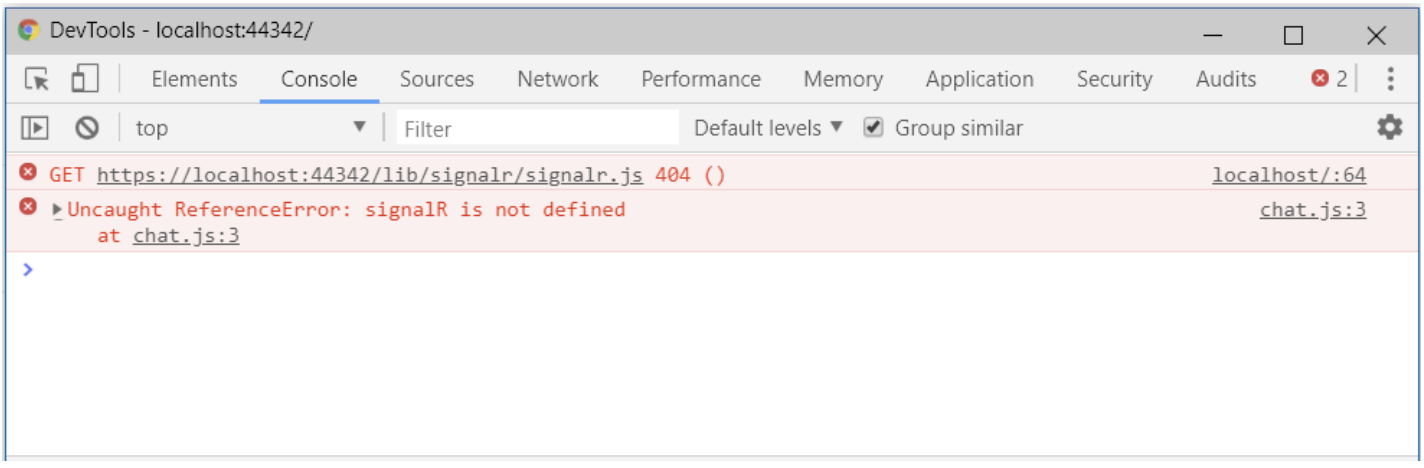
- Choose either browser, enter a name and message, and select the **Send** button.

The name and message are displayed on both pages instantly.



Tip

If the app doesn't work, open your browser developer tools (F12) and go to the console. You might see errors related to your HTML and JavaScript code. For example, suppose you put *signalr.js* in a different folder than directed. In that case the reference to that file won't work and you'll see a 404 error in the console.



Next steps

If you want clients to connect to a SignalR app from different domains, you have to enable Cross-Origin Resource Sharing (CORS). For more information, see Cross-origin resource sharing.

To learn more about SignalR, hubs, and JavaScript clients, see these resources:

- Introduction to SignalR for ASP.NET Core
- Use hubs in SignalR for ASP.NET Core
- ASP.NET Core SignalR JavaScript client