

LEARNING SIGNALR

Comprehensive Guide To Learn SignalR.

Table of Contents:

1. Introduction:

- Overview of Real-Time Web Communication.
- Importance of SignalR in Web Development.
- Introduction to SignalR.

2. Prerequisites:

- Basic Understanding of Web Development.
- Set Up a Development Environment.
- .NetCore and Angular understanding (Note: Demo purpose only).

3. SignalR Overview

- Transport Type
- Properties and method.

4. Setting up SignalR

- Installing SignalR
- Creating a SignalR Hub
- Integrating SignalR with Frontend Technologies

5. Understanding SignalR Architecture

- Connection Overview
- Hubs and Clients

6. Customization

- Creating Custom Hubs
- Setting Up SignalR Events

7. Best Practices

- Performance Optimization
- Code Organization and Structure
- Handling Connection Interruptions

8. Resources

- Official SignalR Documentation
- Online Tutorials and Community Forums

9. Demo Project

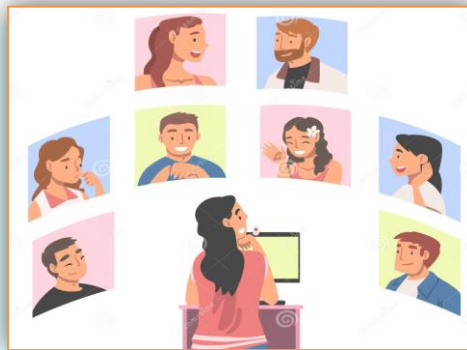
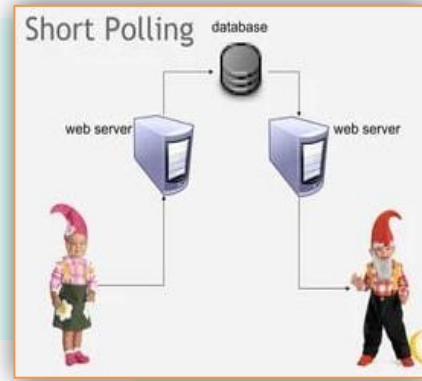
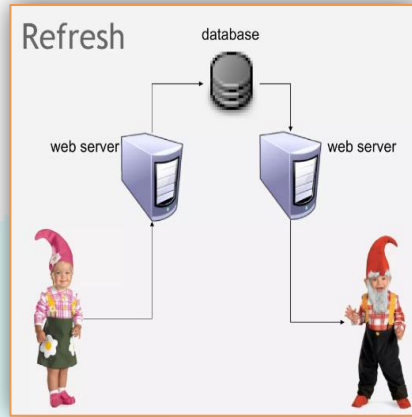
- Creating a Real-Time Vehicle Application

- ✓ **Created Under guidance of Nilesh Prajapati.**
- ✓ **Level of Document: Basic.**
- ✓ **Outcome: Basic understanding of SignalR and able to create a basic project.**

LEARNING SIGNALR

1. Introduction:

1.1 Overview of Real-Time Web Communication:



Importance of Real-Time Communication

- ✓ Enhanced User Engagement
- ✓ Efficient Collaboration
- ✓ Live Updates and Notifications
- ✓ Interactive User Interfaces

Impact on User Experience:

- ✓ Reduced Latency:
- ✓ Increased Interactivity:
- ✓ Timely Feedback
- ✓ Live Collaboration

In conclusion, embracing real-time communication on the web is pivotal for creating user-centric applications that cater to the evolving expectations of today's audience. Whether in social media, business tools, or interactive websites, the impact of real-time communication resonates across various digital experiences, shaping a more responsive and engaging online world.

1.2 Importance of SignalR in Web Development

One powerful tool that facilitates this level of interactivity is SignalR, a library for building real-time applications with ASP.NET and .NET Core. SignalR enriches web applications by enabling seamless, bidirectional communication between the server and clients, fostering a more engaging and responsive environment.

Exploring the Role of SignalR:

- ✓ Real-Time Bidirectional Communication
- ✓ Enhanced User Engagement
- ✓ Live Collaboration
- ✓ Push Notifications
- ✓ Dynamic Content Synchronization

Benefits for Interactive Web Applications:

- ✓ Real-Time Dashboards
- ✓ Live Chat and Messaging
- ✓ Online Gaming
- ✓ Stock Ticker and Financial Updates.
- ✓ Live Polling and Voting

In essence, SignalR stands as a pivotal tool for developers seeking to elevate the interactivity and responsiveness of their web applications. By incorporating SignalR, developers can craft dynamic, real-time experiences that captivate users and align with the evolving expectations of modern web interactions..

1.3 Introduction to SignalR

- ✓ SignalR stands out as a powerful and versatile library designed for building real-time applications within the ASP.NET and .NET Core ecosystems. Introduced by Microsoft, SignalR simplifies the complexities of real-time communication by providing developers with a seamless framework to implement interactive features in web applications.
- ✓ Library for implementation of realtime communication.
- ✓ It wrapper over 3 transport protocol:
 - Web Socket
 - Long Polling
 - Server Sent Event (SSE)

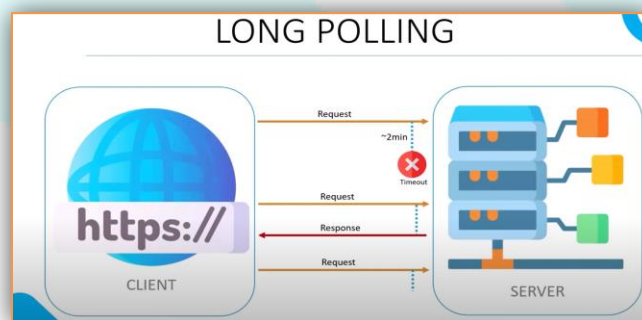
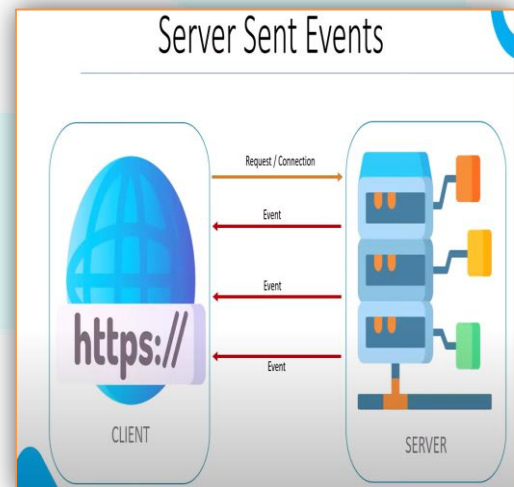
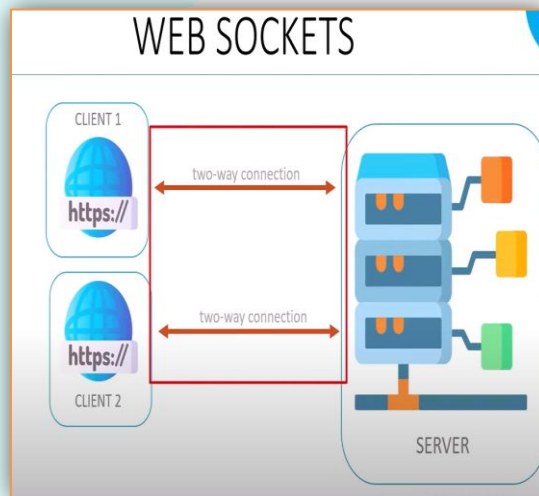
2. Prerequisites:

- ✓ Basic Understanding of Web Development.
- ✓ Set Up a Development Environment.
- ✓ Protocol Http request and response.
- ✓ Client server Architecture.

LEARNING SIGNALR

3. SignalR Overview:

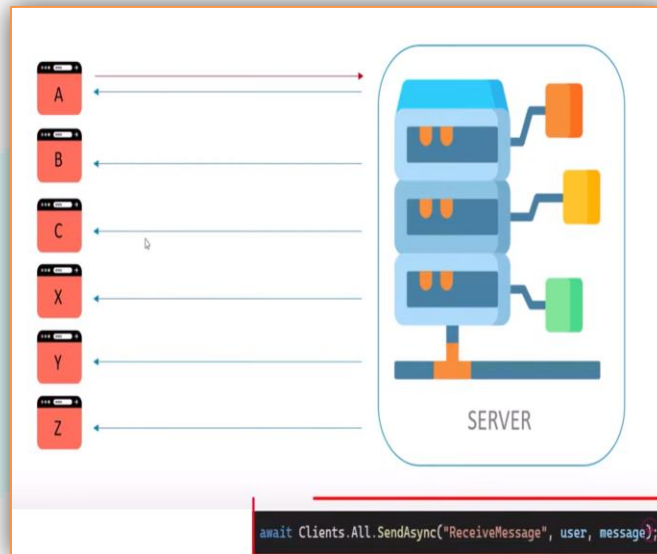
3.1 Transport Type.



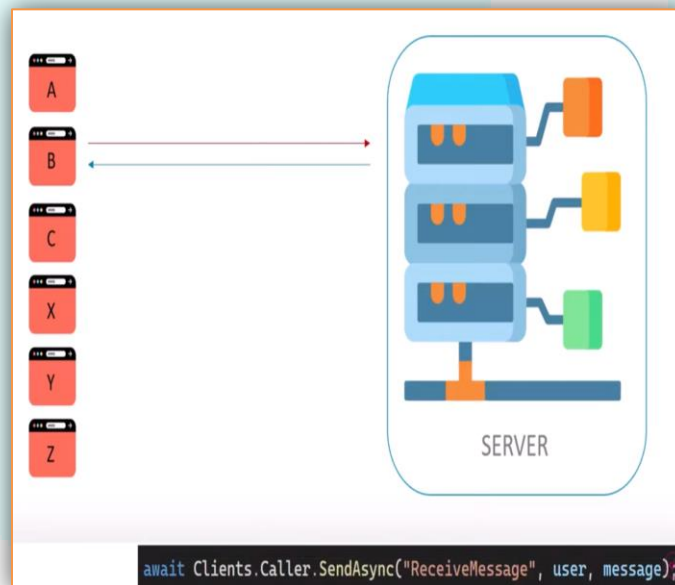
LEARNING SIGNALR

3.2 Properties and Method.

- ✓ Send data to all client

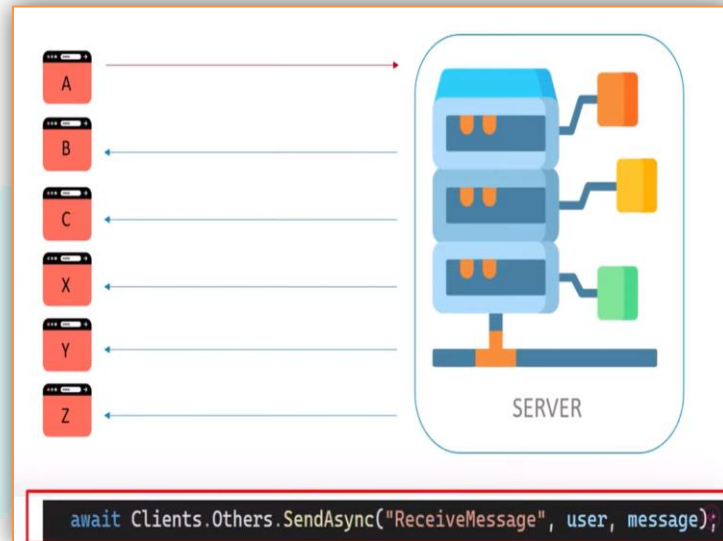


- ✓ Send data to caller client.

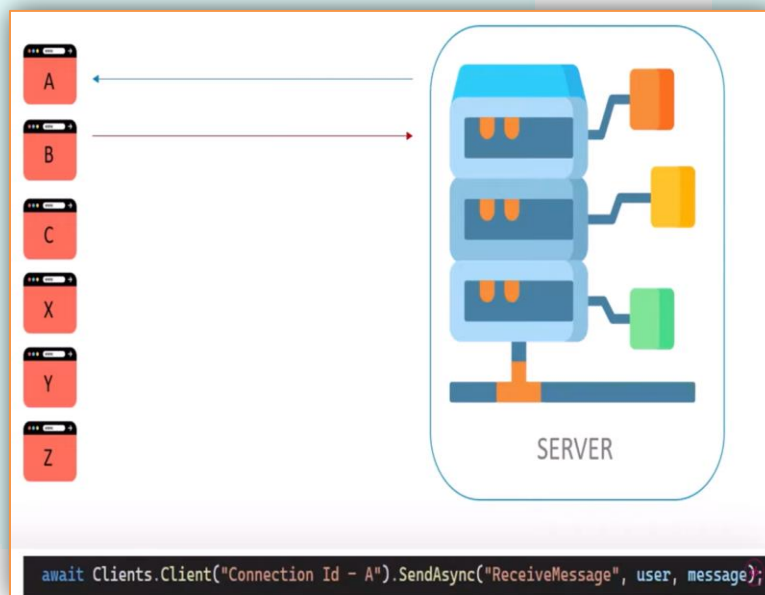


LEARNING SIGNALR

- ✓ Send data to all client except caller client.

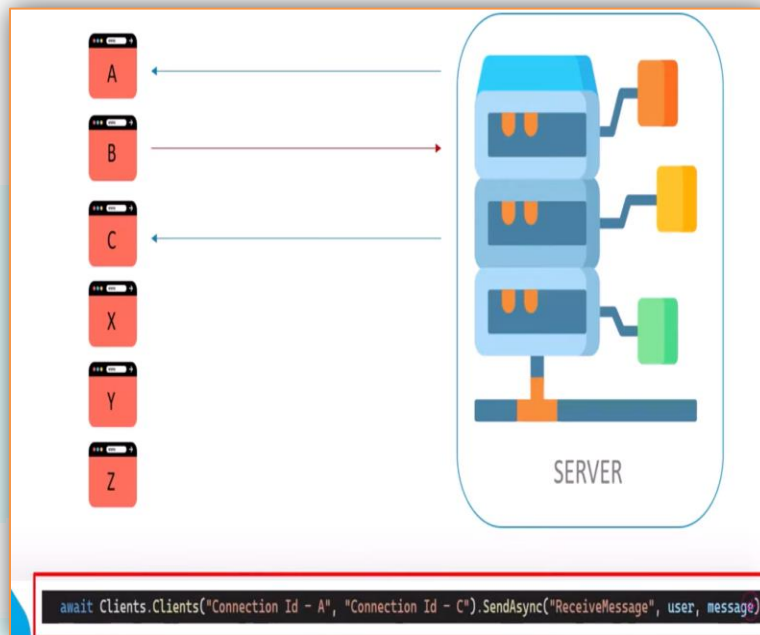


- ✓ Send data to specific client (Tip : Context.ConnectionId give the user connection id of current caller client)

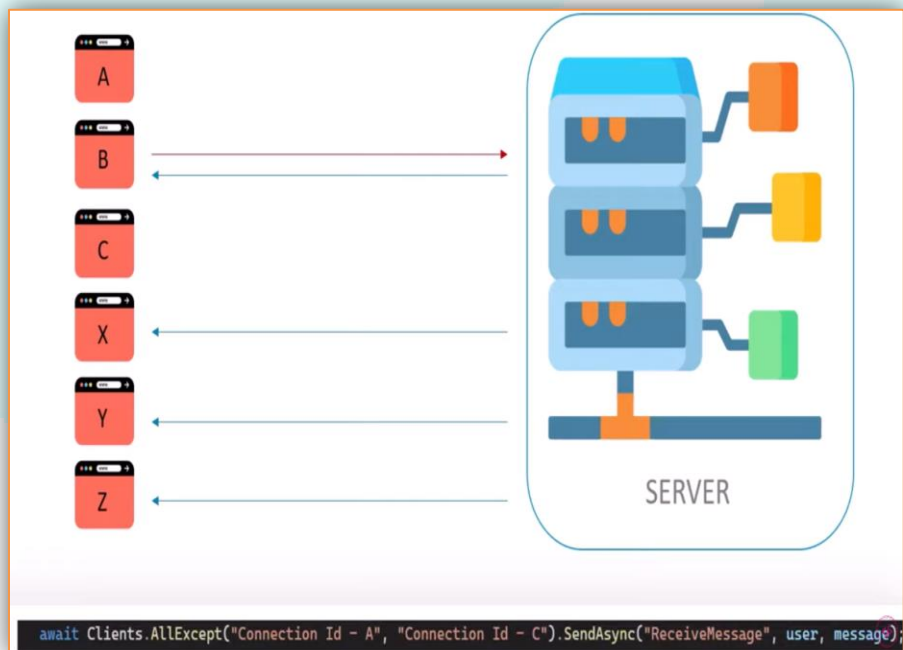


LEARNING SIGNALR

- ✓ Send data to specific clients.



- ✓ Send data to all client but except some client.



LEARNING SIGNALR

✓ Send and Invoke method in signalR.

Snippet code :hubconnection.send(),hubconnection.invoke()

- Send returns a promise that is resolved when the client has sent the invocation to the server, or an error occurred. The server may still be handling the invocation when the promise resolves.
- Invoke returns a promise that is resolved when the server has finished invoking the method (or an error occurred). In addition, the Invoke promise can receive a result from the server method, if the server returns a result.

SignalR provides several default functions and methods that can be used on both the server and client sides. Here are some of the key default functions of SignalR:

✓ Server-Side Default Functions:

- ✓ OnConnectedAsync() and OnDisconnectedAsync(Exception exception)
 - **OnConnectedAsync:** This method is called when a new client connects to the hub.
 - **OnDisconnectedAsync:** Called when a client disconnects, either voluntarily or due to an exception.

```
public override async Task OnConnectedAsync()
{
    // Code to execute when a client connects.
}

public override async Task OnDisconnectedAsync(Exception exception)
{
    // Code to execute when a client disconnects.
}
```

- ✓ OnReconnected():
 - Called when a client that was previously disconnected successfully reconnects.

```
public override async Task OnReconnected()
{
    // Code to execute when a client successfully reconnects.
}
```


LEARNING SIGNALR

Client-Side Default Functions:

- ✓ `start()` and `stop()`:
- ✓ `start()`: Initiates the connection to the hub.
- ✓ `stop()`: Closes the connection to the hub.

```
const connection = new signalR.HubConnectionBuilder()
    .withUrl("/hub")
    .build();

connection.start().then(() => {
    console.log("Connection started");
}).catch(err => console.error(err));
```

- ✓ state:

Represents the state of the connection, providing information such as whether the connection is connected or disconnected.

```
console.log(connection.state);
```

- ✓ Possible values: "Connecting", "Connected", "Reconnecting", "Disconnected"

ServerSide Code

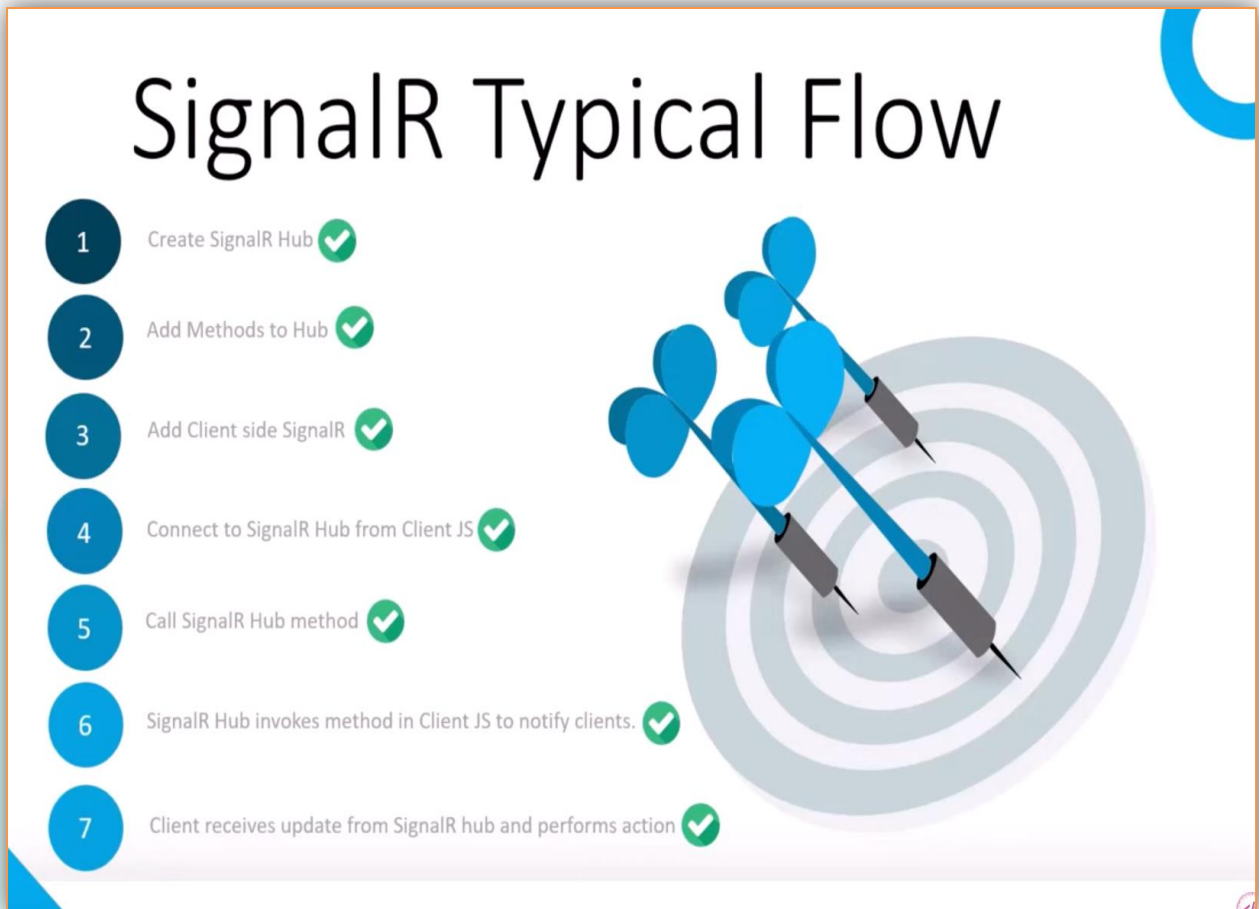
```
public class MyHub : Hub
{
    public void SendMessage(string user, string message)
    {
        Clients.All.SendAsync("ReceiveMessage", user, message);
    }
}
```

Client Side Code

```
connection.invoke("SendMessage", user, message)
    .then(() => {
        console.log("Message sent successfully");
    })
    .catch(err => console.error(err));

connection.on("ReceiveMessage", (user, message) => {
    console.log(`${user}: ${message}`);
});
```

4. Setting up SignalR:



4.1 Installing SignalR

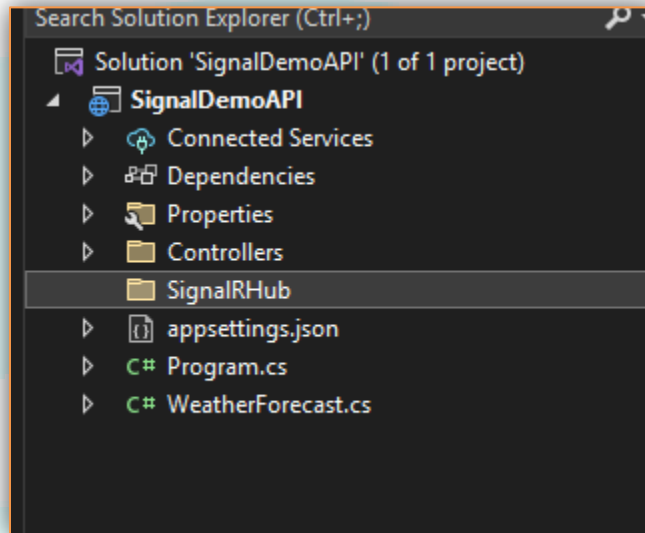
- ✓ Considering you have idea of creating .net core web Api project.([Click me to learn about how to create me.](#))
- ✓ For angular side and .net core api has inbuilt library for signalr so no need to install from nuget package.

```
npm install @microsoft/signalr
```

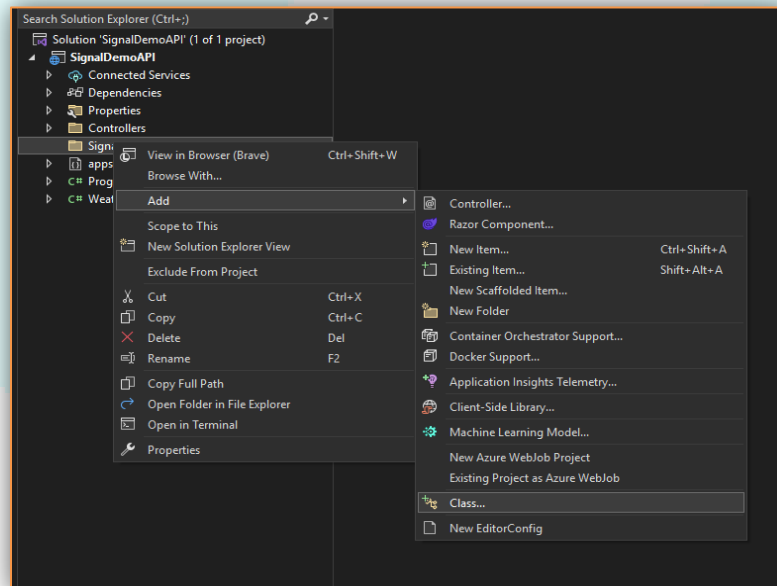
LEARNING SIGNALR

4.2 Creating a SignalR Hub

- ✓ Create a folder in your webapi .net core project with name SignalRHub.

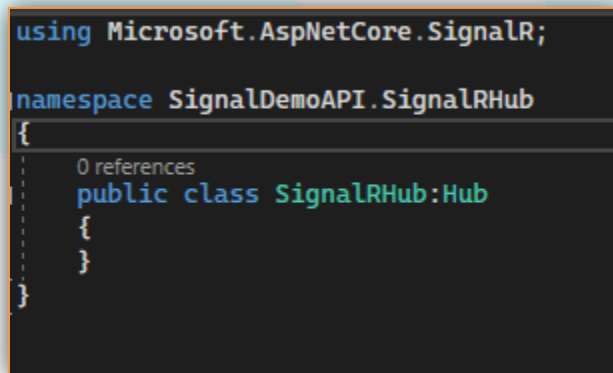
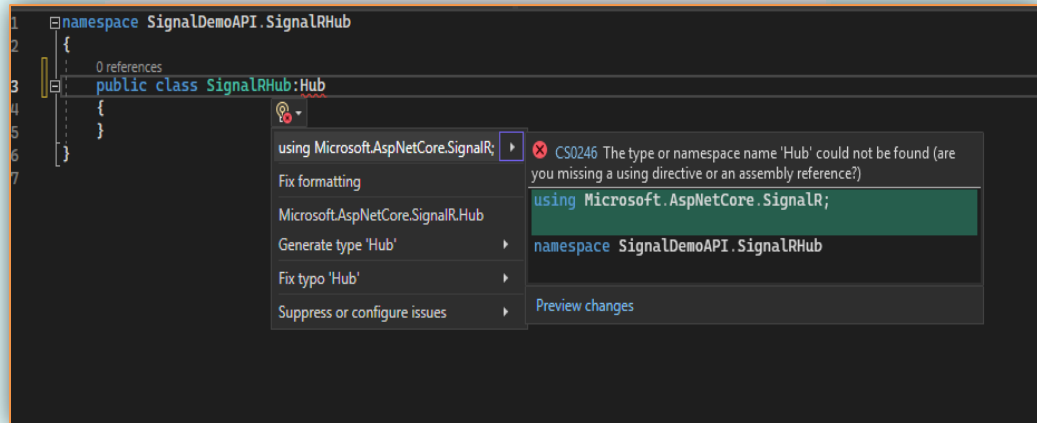
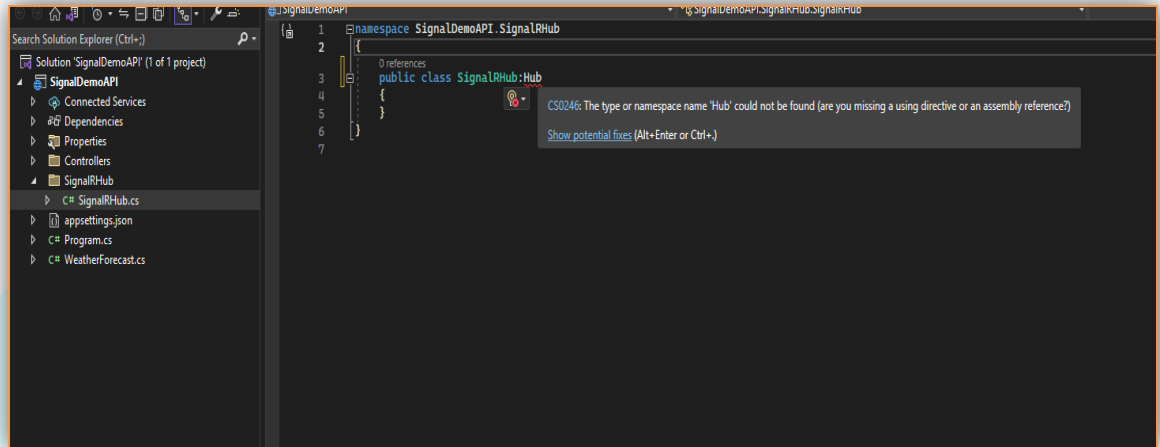


- ✓ Create a class in that folder with name SignalRHub.



LEARNING SIGNALR

- ✓ Inherit the SignalRHub class with Hub class from namespace call Microsoft.AspNetCore.SignalR.



LEARNING SIGNALR

```
using Microsoft.AspNetCore.SignalR;  
  
namespace SignalDemoAPI.SignalRHub  
{  
    0 references  
    public class SignalRHub:Hub  
    {  
        //first method to call when Client connects.  
        0 references  
        public override Task OnConnectedAsync()  
        {  
            Console.WriteLine(Context.ConnectionId); // Id of Connected Client  
            return base.OnConnectedAsync();  
        }  
  
        0 references  
        public override Task OnDisconnectedAsync(Exception? exception)  
        {  
            return base.OnDisconnectedAsync(exception);  
        }  
  
        0 references  
        public async Task NewMessage(string msg)  
        {  
            await Clients.All.SendAsync("MessageReceived", msg);  
        }  
    }  
}
```

- ✓ Created NewMessage Method method in Hub .
- ✓ Now its time to register SignalR services in Program.cs file.

```
Program.cs*  SignalRHub.cs  
SignalDemoAPI  
{  
    1 var builder = WebApplication.CreateBuilder(args);  
    2  
    3 // Add services to the container.  
    4  
    5 builder.Services.AddControllers();  
    6 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle  
    7 builder.Services.AddEndpointsApiExplorer();  
    8 builder.Services.AddSignalR(); // Add signalR service  
    9 builder.Services.AddSwaggerGen();  
    10  
    11 var app = builder.Build();  
    12  
    13 // Configure the HTTP request pipeline.  
    14 if (app.Environment.IsDevelopment())  
    15 {  
    16     app.UseSwagger();  
    17     app.UseSwaggerUI();  
    18 }  
    19  
    20 app.UseHttpsRedirection();  
    21  
    22 app.UseAuthorization();  
    23  
    24 app.MapControllers();  
    25  
    26 app.Run();  
    27
```

LEARNING SIGNALR

- ✓ Add cors policy so that our client will be able to connect the hub and server.

```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSignalR(); // Add SignalR service
builder.Services.AddCors(options =>
{
    options.AddPolicy("CorsPolicy",
        builder =>
        {
            builder.WithOrigins("http://localhost:4200")
                .AllowAnyMethod()
                .AllowCredentials()
                .AllowAnyHeader();
        });
}); // Add Cors

// Add Swagger for API documentation
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "SignalDemo API", Version = "v1" });
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseCors("CorsPolicy"); // Add here
app.UseAuthorization();

app.MapControllers();

app.Run();
```

- ✓ Now let add end point of our hub.

```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSignalR(); // Add SignalR service
builder.Services.AddCors(options =>
{
    options.AddPolicy("CorsPolicy",
        builder =>
        {
            builder.WithOrigins("http://localhost:4200")
                .AllowAnyMethod()
                .AllowCredentials()
                .AllowAnyHeader();
        });
}); // Add Cors

// Add Swagger for API documentation
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "SignalDemo API", Version = "v1" });
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseCors("CorsPolicy");
app.UseRouting(); // Add here
app.UseAuthorization();

app.UseEndpoints(endpoints =>
{
    endpoints.MapHub<SignalRHub>("/SignalRHub");
});

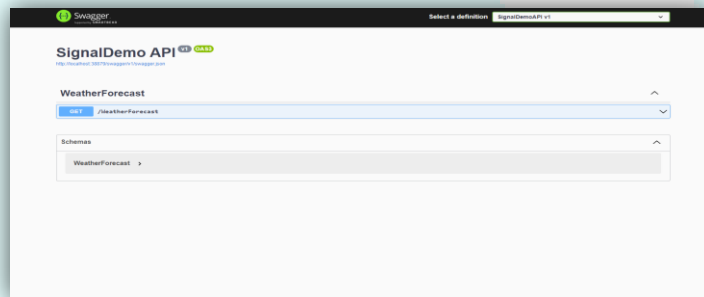
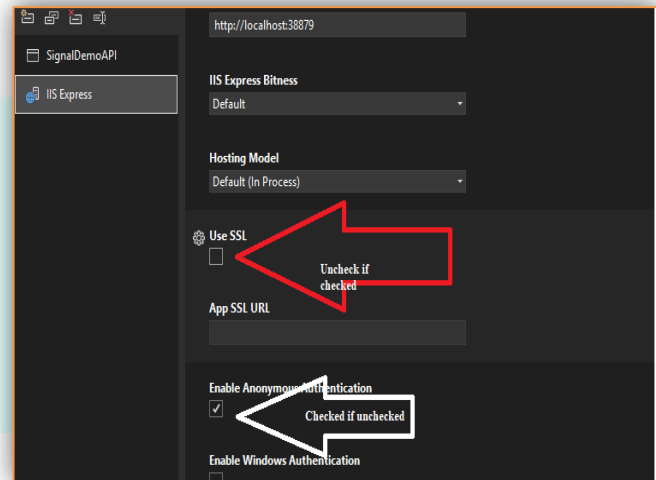
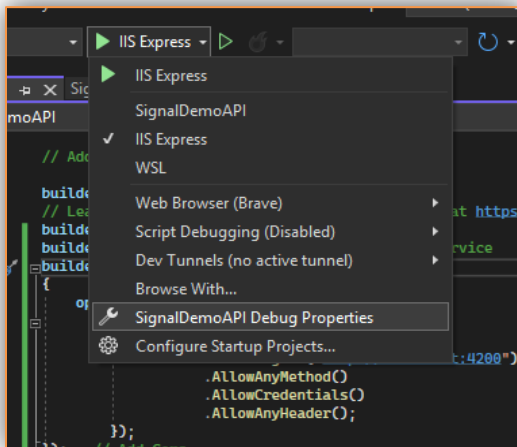
app.MapControllers();
```

Hub Class Name

Name should in client call I will mention while client side writing

LEARNING SIGNALR

Before we go client side code we have to check two things in VS code .



LEARNING SIGNALR

4.3 Integrating SignalR with Frontend Technologies: (Angular)

- ✓ Considering you have idea of how to create angular project.([Click me to know how to create angular project.](#))
- ✓ Now let create a service in angular project with name signalrservice.

```
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide\src\app> ng g s signalrservice
CREATE src/app/signalrservice.service.spec.ts (413 bytes)
CREATE src/app/signalrservice.service.ts (152 bytes)
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide\src\app> ng serve
```

Initial Chunk Files	Names	Raw Size
polyfills.js	polyfills	83.46 kB
main.js	main	23.18 kB
styles.css	styles	96 bytes
Initial Total		106.74 kB

```
Application bundle generation complete. [2.422 seconds]
Watch mode enabled. Watching for file changes...
  → Local: http://localhost:4200/
  → press h + enter to show help
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide\src\app>
```

- ✓ Installing SignalR in angular project.

```
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide> npm install @microsoft/signalr

added 16 packages, and audited 928 packages in 19s

119 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide>
```

- ✓ Now its time to work on signalrservice and write client side code to connect with hub.(SignalRHub in our case).
- ✓ Steps to be in mind:
 - CreateConnection→StartConnection→RegisterEvent.
 - Creating object of HubConnectionBuilder.
 - Basic recap of on,start,invoke...

LEARNING SIGNALR

```
@Injectable()
export class SignalRServiceService {
  messageReceived = new EventEmitter<string>();
  connectionEstablished = new EventEmitter<Boolean>();
  private _hubConnection!: HubConnection;
  private connectionIsEstablished = false;

  createConnection() {
    this._hubConnection = new HubConnectionBuilder()
      .withUrl('http://localhost:38879/SignalRHub')
      .build();
  }
}
```

Step 1

Step 2



```
app.UseHttpsRedirection();
app.UseCors("CorsPolicy");
app.UseRouting();
app.UseAuthorization();
app.UseEndpoints(endpoints => {
  endpoints.MapHub<SignalRHub>("/SignalRHub");
});
```

```
startConnection(): void {
  this._hubConnection
    .start()
    .then(() => {
      this.connectionIsEstablished = true;
      console.log('Hub connection started');
      this.connectionEstablished.emit(true);
    })
    .catch(err => {
      console.log('Error while establishing connection, retrying...');
      setTimeout(() => { this.startConnection(); }, 5000);
    });
}
```

Step 3

```
registerOnServerEvents(): void {
  this._hubConnection.on('MessageReceived', (data: any) => {
    this.messageReceived.emit(data);
  });
}
```

LEARNING SIGNALR

```
import { HubConnection, HubConnectionBuilder } from '@microsoft/signalr';

@Injectable()
export class SignalrServiceService {

  messageReceived = new EventEmitter<string>();
  connectionEstablished = new EventEmitter<Boolean>();
  private _hubConnection!: HubConnection;
  private connectionIsEstablished = false;
  constructor()
  {
    this.createConnection();
    this.startConnection();
    this.registerOnServerEvents();
  }
}
```

Add in constructor of SignalrService
Step 4

- ✓ Next step is to set up our landing page and call this signalrService method and connect to our hub.
- ✓ Considering basic level documentation I had created all things in app component only you can also do it by creating new page and use routing.

```
1 import { Component, NgZone } from '@angular/core';
2 import { SignalrServiceService } from './signalrService.service';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.scss']
8 })
9 export class AppComponent {
10   title = 'SignalR Demo Client Side';
11   txtMessage: string = '';
12   messages = new Array<string>();
13
14   constructor(
15     private messageService: SignalrServiceService,
16     private _ngZone: NgZone,
17   ) {
18     this.subscribeToEvents();
19   }
20
21   MessageSet(event:any)
22   {
23     console.log(event);
24     this.txtMessage=event.target.value;
25   }
26
27   sendMessage(): void {
28     if (this.txtMessage) {
29       this.txtMessage = '';
30     }
31   }
32
33   private subscribeToEvents(): void {
34     this.messageService.messageReceived.subscribe((message: string) => {
35       this._ngZone.run(() => {
36         this.messages.push(message);
37       });
38     });
39   }
40 }
41
42
43
```

Step 1

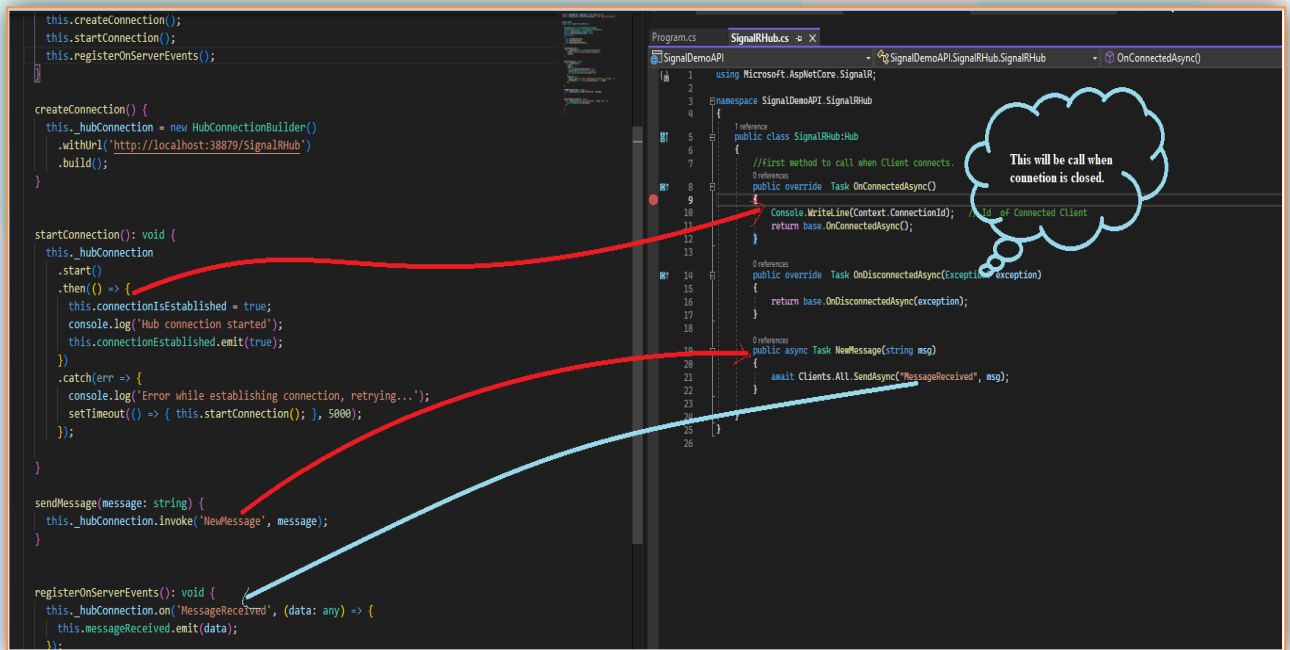
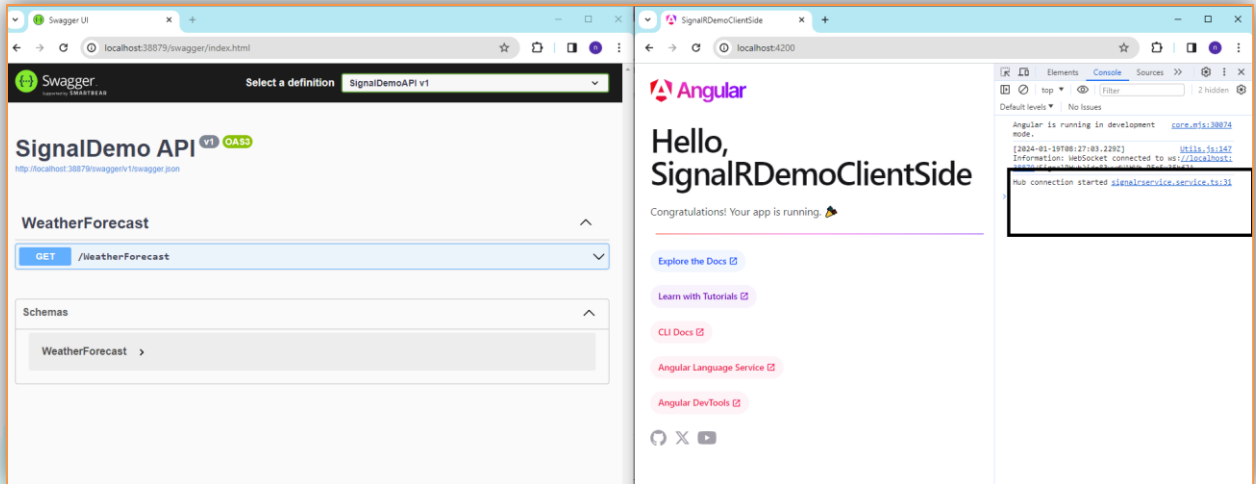
Step 3

Step 2

LEARNING SIGNALR

5. Understanding SignalR Architecture:

5.1 Connection Overview



LEARNING SIGNALR

5.2 Hubs and Clients

The screenshot displays two code files in VS Code. On the left is `AppComponent.ts`, which defines a client-side component with a `MessageSet` event and a `sendMessage` method. A handwritten orange cloud labeled "Client" points to this file. On the right is `SignalRHub.cs`, which defines a server-side hub with methods `OnConnectedAsync`, `OnDisconnectedAsync`, and `SendMessage`. A handwritten orange cloud labeled "Hub" points to this file.

6. Customization:

6.1 Creating Custom Hubs

- ✓ Override the hub function and can also create new hub in the same server as shown above and have to add endpoint similar to individual.

6.2 Setting Up SignalR Events

This screenshot shows a more detailed view of the SignalR setup. It includes the `AppComponent.ts` file on the left, the `SignalService.cs` file in the middle, and the `SignalRHub.cs` file on the right. Numbered annotations are present: 1 points to the `sendMessage` method in `AppComponent.ts`; 2 points to the `startConnection` method in `SignalService.cs`; 3 points to the `SendMessage` method in `SignalRHub.cs`; and 4 points to the `MessageSet` event in `AppComponent.ts`.

LEARNING SIGNALR

- ✓ Now you are done with basic project.
- ✓ Start your Server(.net core api) and your client (angular) in multiple browser and boom!!!!

The image displays four browser windows arranged in a 2x2 grid, each showing a SignalR client interface. The top-left window is titled 'SignalR' and shows a 'Send' button and a 'Message Box' containing a list of messages: 'Nilesh this side', 'Manav this side', 'Hemang this side', and 'Yash this side'. The top-right window is titled 'SignalR Clients' and shows a 'Send' button and a 'Message Box' containing the same list of messages. The bottom-left window is titled 'SignalR Clients' and shows a 'Send' button and a 'Message Box' containing the same list of messages. The bottom-right window is titled 'SignalR Clients' and shows a 'Send' button and a 'Message Box' containing the same list of messages. Each window has a browser address bar showing 'localhost:4200'.



7. Best Practices:

- **Performance Optimization**
 - If possible use primary database like redis.
- **Code Organization and Structure**
- **Handling Connection Interruptions**

8. Resources:

8.1 Official SignalR Documentation

- ✓ [Official Docs of microsoft of SignalR.](#)

8.2 Online Tutorials and Community Forums

- ✓ [Click to learn me through video.](#)

9. Demo Project:

- ✓ [Click for Demo project](#)

Note: If any suggestion or any changes in demo please go ahead new suggestion are always welcome..



