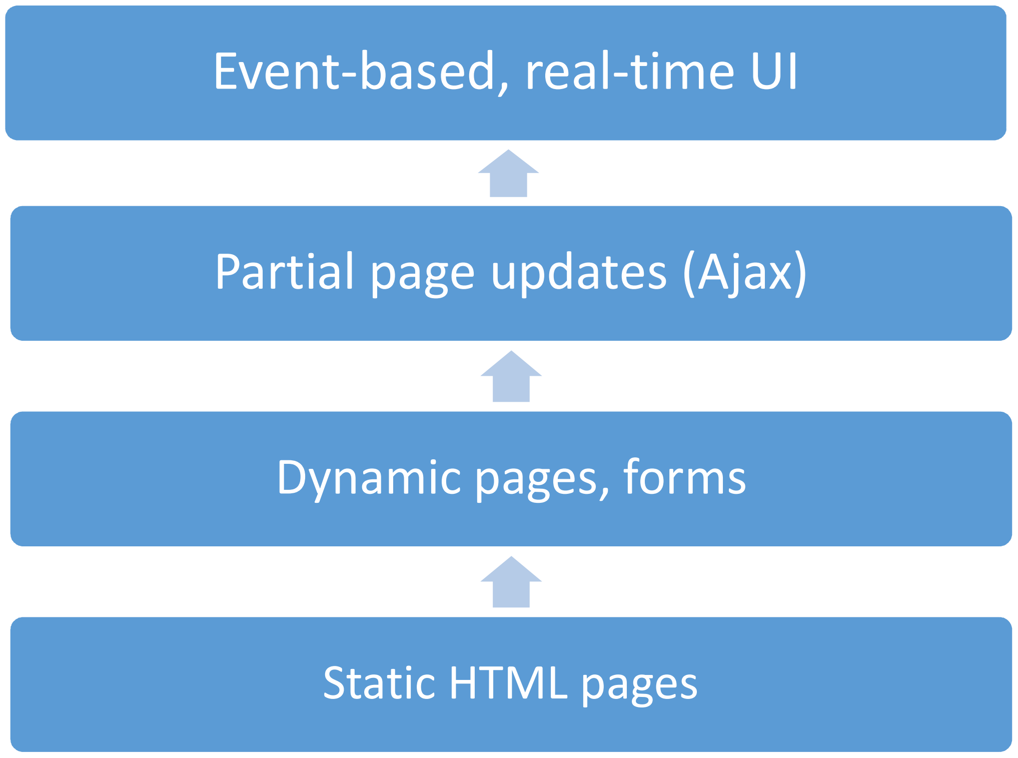
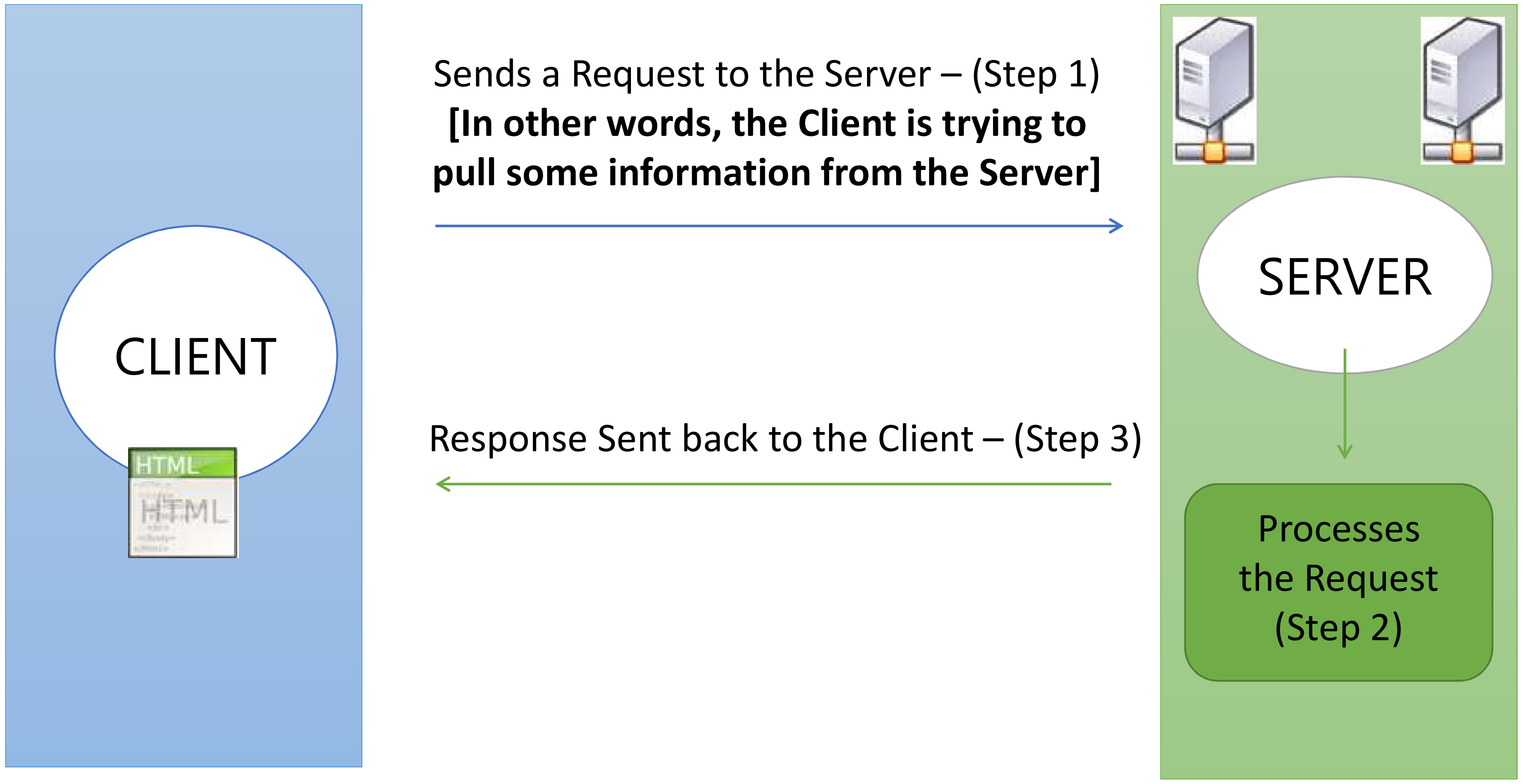
# SignalR Core

Welcome to the Real-time world of web

## Web Evolution



# Traditional Web Approach


CLIENT
HTML
HTML
Sends a Request to the Server – (Step 1)
**[In other words, the Client is trying to pull some information from the Server]**
SERVER
Processes the Request (Step 2)
Response Sent back to the Client – (Step 3)

## What is Real Time Web Application?

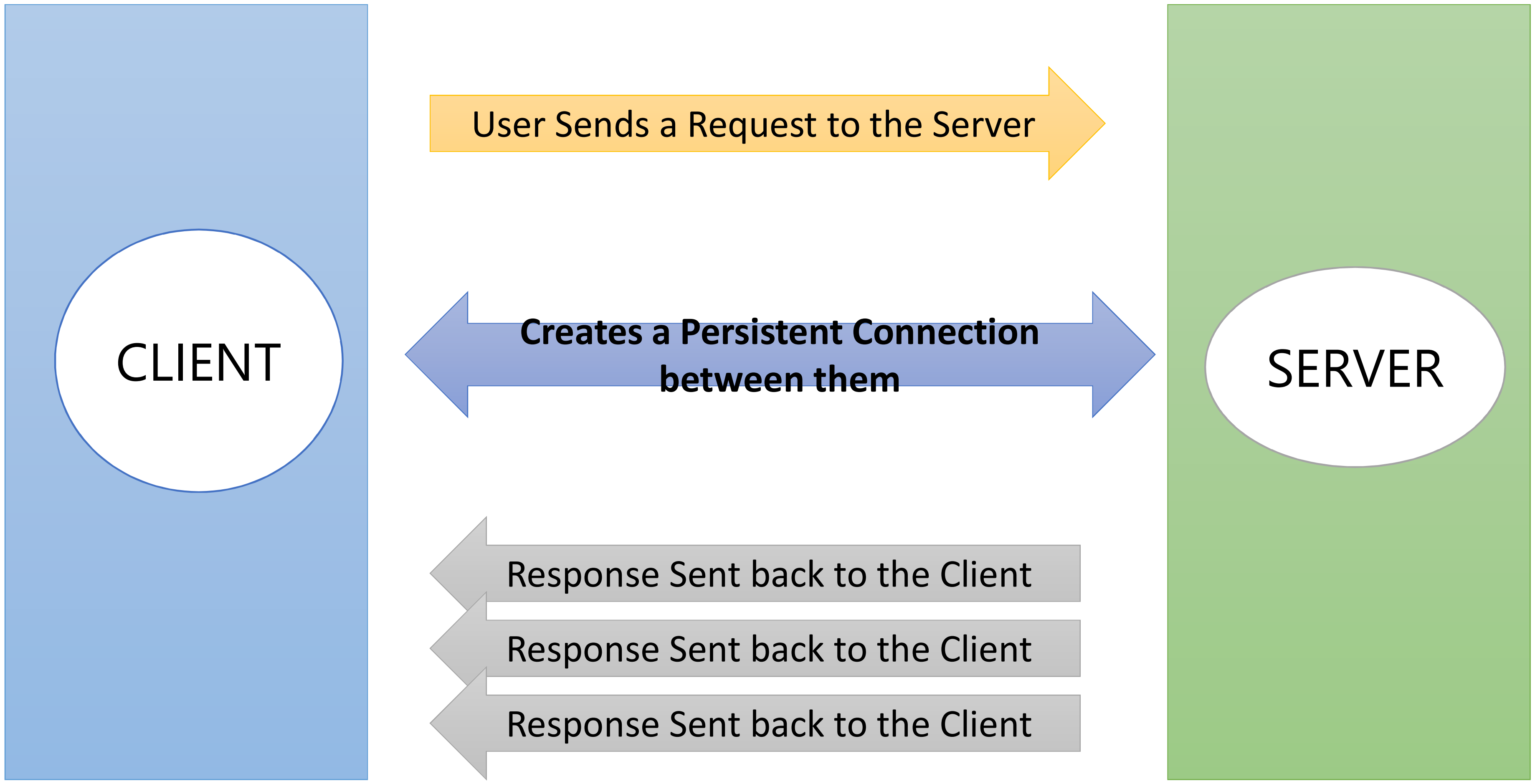
- In simple terms, “Real Time” means an immediate response being sent by the Server to the Client.

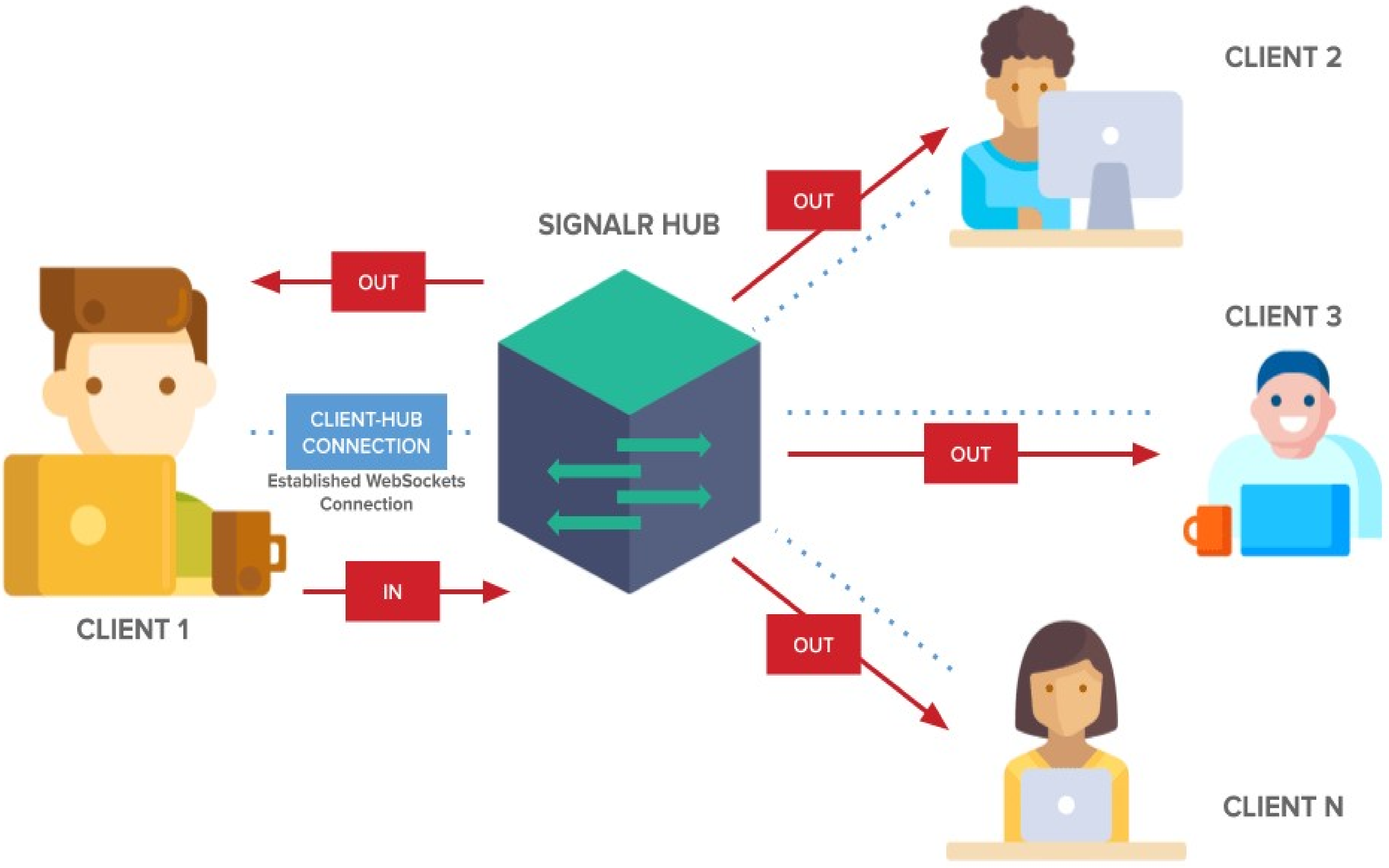
- Real Time is all about “Pushing” instead of “Pulling”

- Push Technology is completely different from Pull Technology. Its about getting told what’s new, instead of asking for what’s new!!!

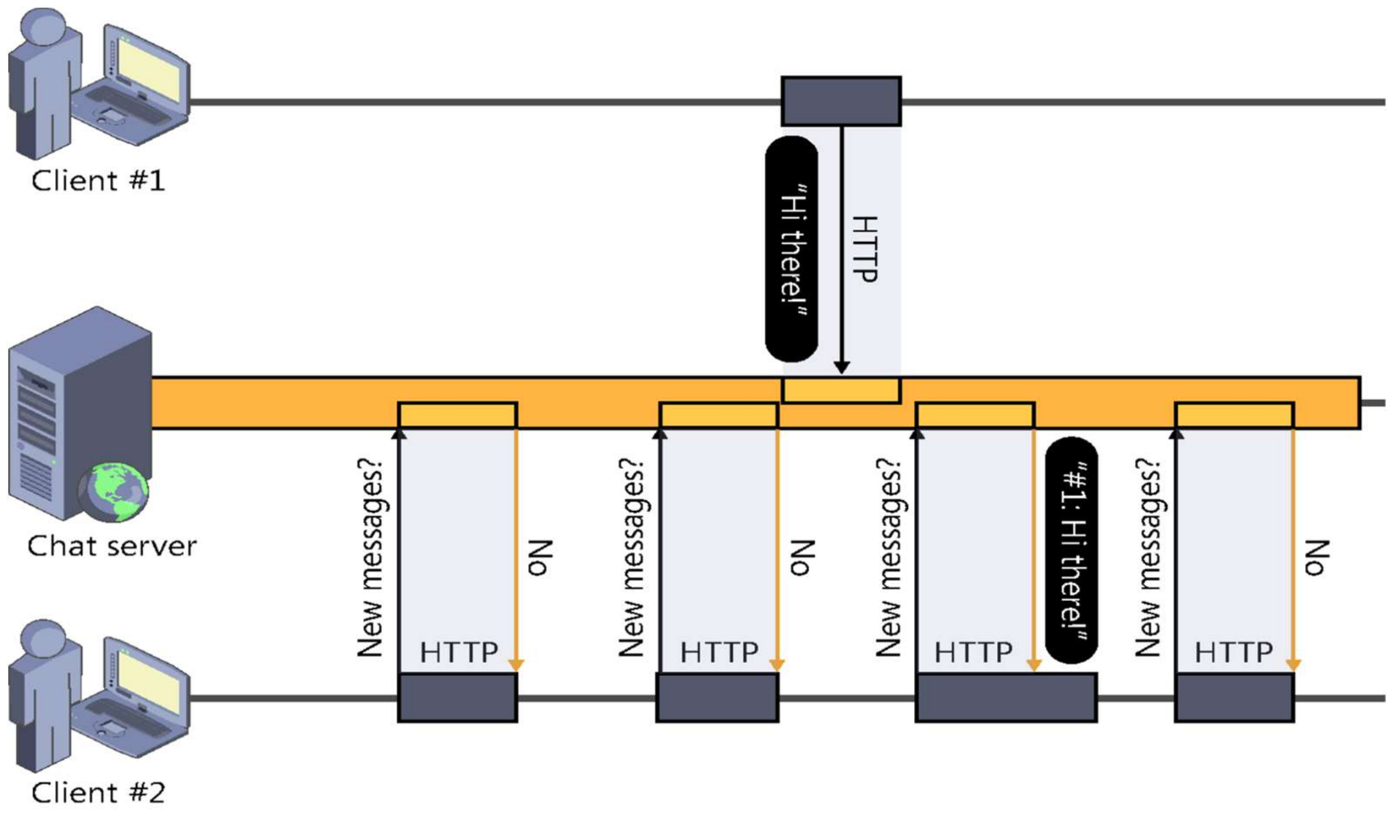
- Facebook, Twitter, Yahoo Cricket Live, Stock Ticker

## Real Time Web Approach

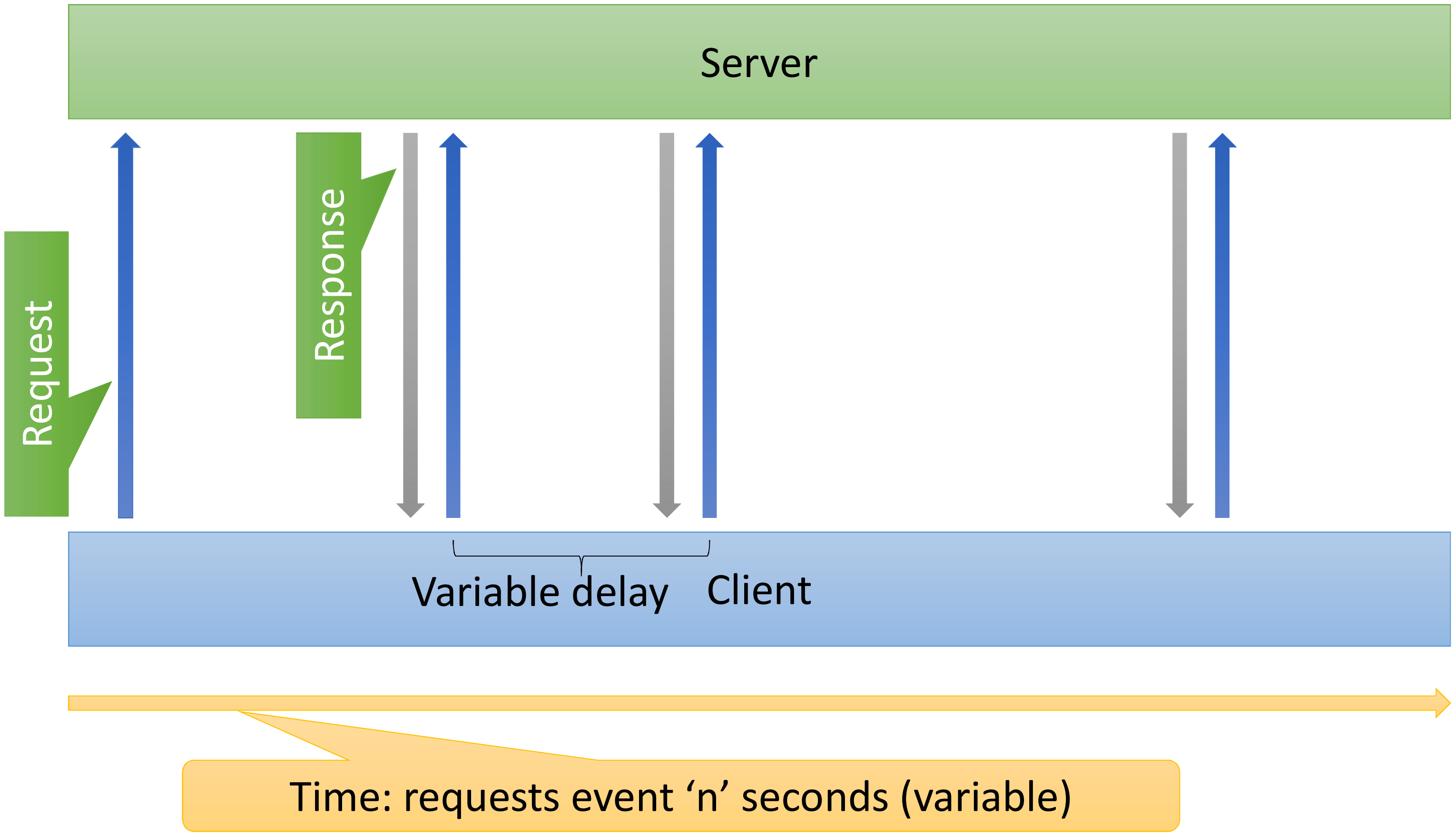

CLIENT
SERVER
User Sends a Request to the Server
Creates a Persistent Connection between them
Response Sent back to the Client
Response Sent back to the Client
Response Sent back to the Client


SIGNALR HUB
CLIENT 1
OUT
CLIENT-HUB CONNECTION
Established WebSockets Connection
IN
OUT
CLIENT 2
CLIENT 3
OUT
OUT
CLIENT N

## Polling


Client #1
"Hi there!"
HTTP
Chat server
New messages?
HTTP
No
New messages?
HTTP
No
New messages?
HTTP
"#1: Hi there!"
New messages?
HTTP
No
Client #2

# Long Polling



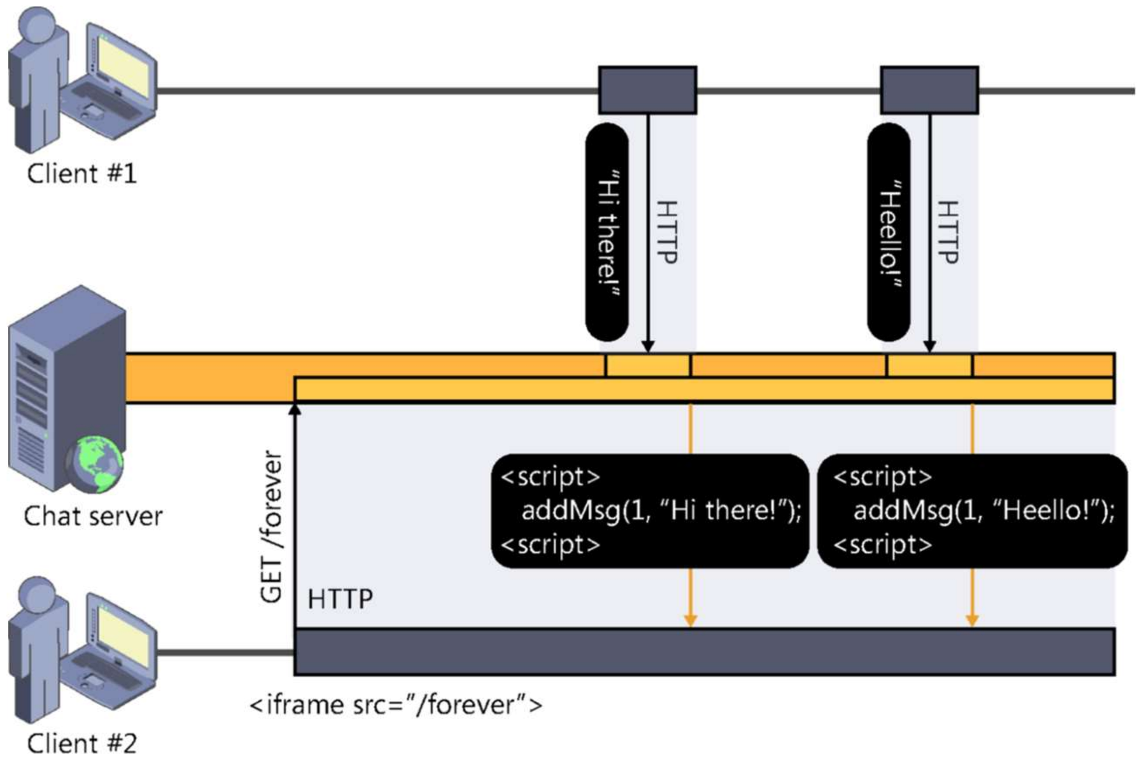
## Forever Frames

- Data is sent out in chunks.
- Chunked Encoding : is the feature in the HTTP 1.1 specification allowing a server to start sending a response before knowing its total length

- A hidden iframe element is opened in the browser after page load, establishing a long-lived connection inside the hidden iframe..

| **Pros** | **Cons** |
|---|---|
| Supported on IE Browser. | ▪ Iframes are loaded again and again with chunks of data.<br>▪ All script tags remain on the page.<br>▪ one-way realtime connection from server to client |
| | |

## Forever Frames



```
<script>
  addMsg(1, "Hi there!");
<script>
```

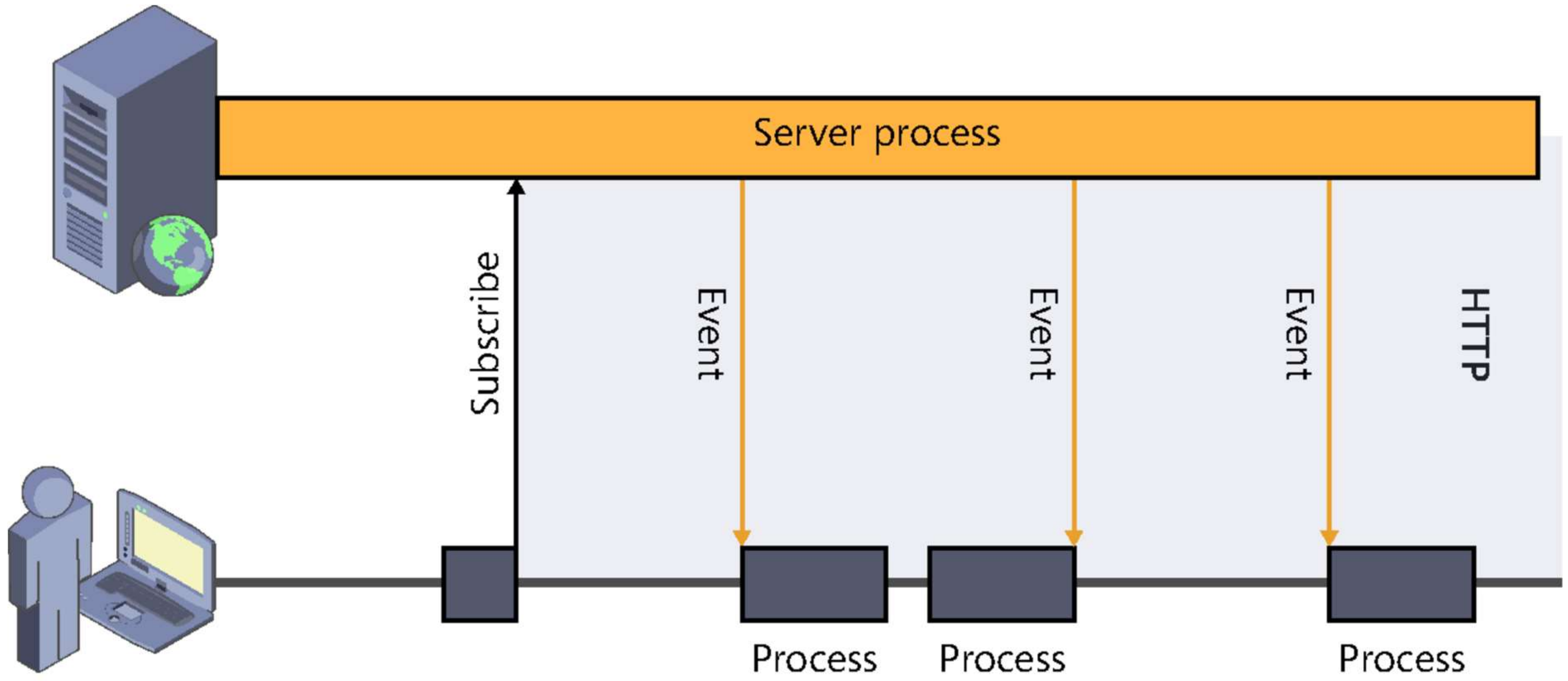
```
<script>
  addMsg(1, "Heello!");
<script>
```

## Server Sent Events

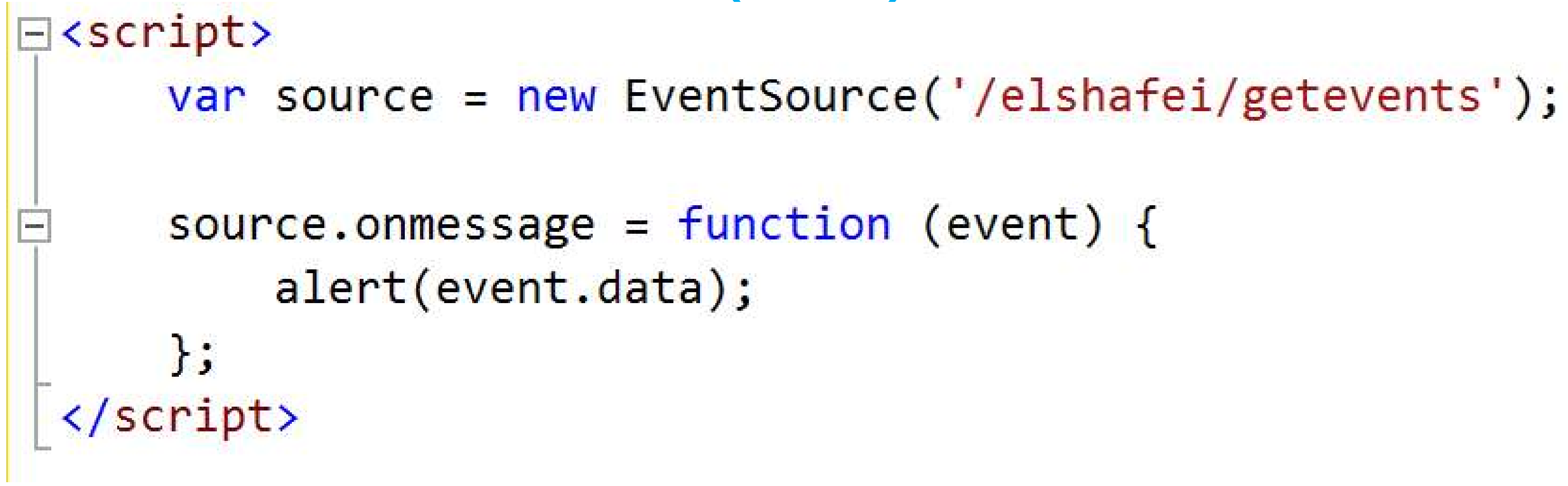
- Requires a single connection between Client-Server.
- Uses Javascript API – “EventSource” through which Client can request a particular URL to receive data stream.
- Used to send Message Notifications or Continuous Data Streams.

| Pros | Cons |
|---|---|
| No need to reconnect | Works in server-to-client direction only |
| | Not supported in IE |

## Server Sent Events(SSE)

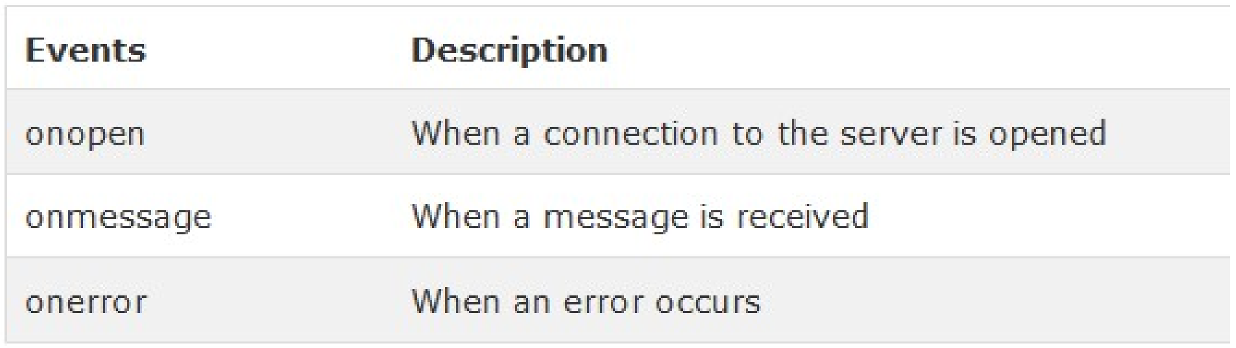


## Server Sent Events(SSE)



```
<script>
    var source = new EventSource('/elshafei/getevents');

    source.onmessage = function (event) {
        alert(event.data);
    };
</script>
```



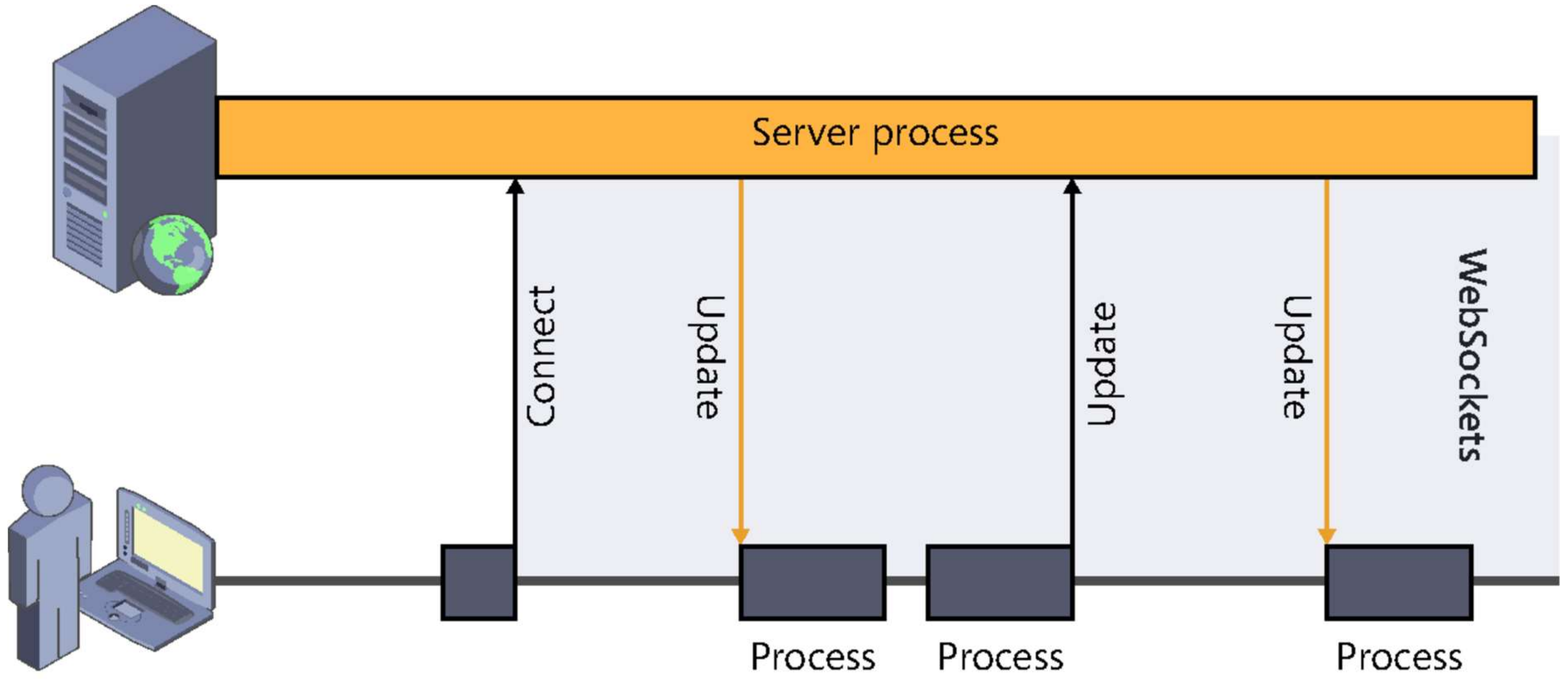
| Events | Description |
|---|---|
| onopen | When a connection to the server is opened |
| onmessage | When a message is received |
| onerror | When an error occurs |

## WebSocket

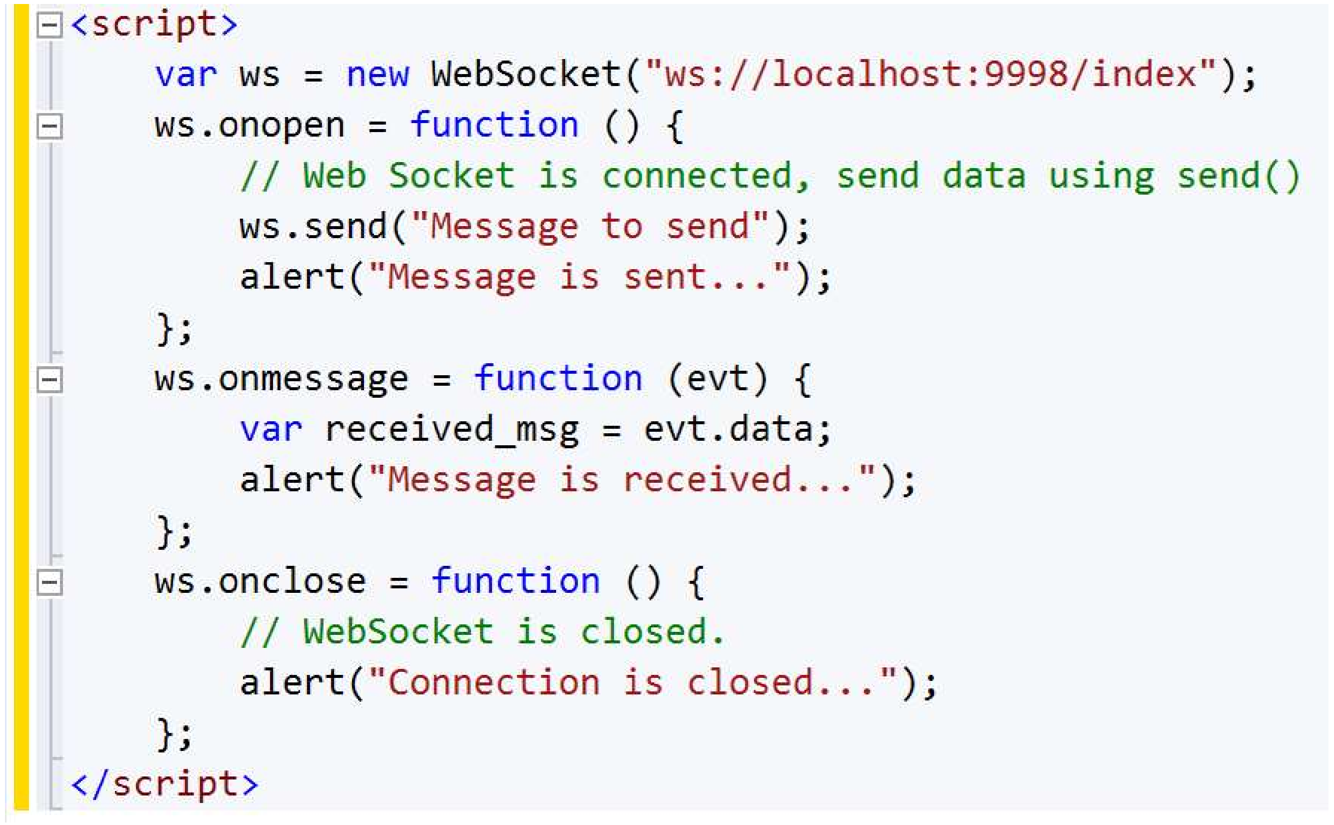
- A new transport technique which came up with HTML5.
- a full-duplex single socket connection over which messages can be sent between client and server
- It internally works on top of TCP protocol.

| **Pros** | **Cons** |
|---|---|
| Full-duplex persistent connection (both ways) | Supported only on latest browsers – (IE 10)<br>Windows 8, Windows Server 2012 or later |
| Fastest solution | Works only with IIS-8.0<br>.Net Framework 4.5+ |

## WebSocket


Server process
Connect
Update
Update
Update
WebSockets
Process
Process
Process

## WebSocket



```
<script>
    var ws = new WebSocket("ws://localhost:9998/index");
    ws.onopen = function () {
        // Web Socket is connected, send data using send()
        ws.send("Message to send");
        alert("Message is sent...");
    };
    ws.onmessage = function (evt) {
        var received_msg = evt.data;
        alert("Message is received...");
    };
    ws.onclose = function () {
        // WebSocket is closed.
        alert("Connection is closed...");
    };
</script>
```

## SignalR – Modern Web

- SignalR is an open-source library that simplifies adding real-time web functionality to apps.
- Concept intiated by “David Fowler” and “Damien Edwards”
- a server-side framework to write push services
- a set of client libraries to make push service communication easy to use on any platform
- optimized for asynchronous processing
- Open Source available on Github!!!

## Good candidates for SignalR

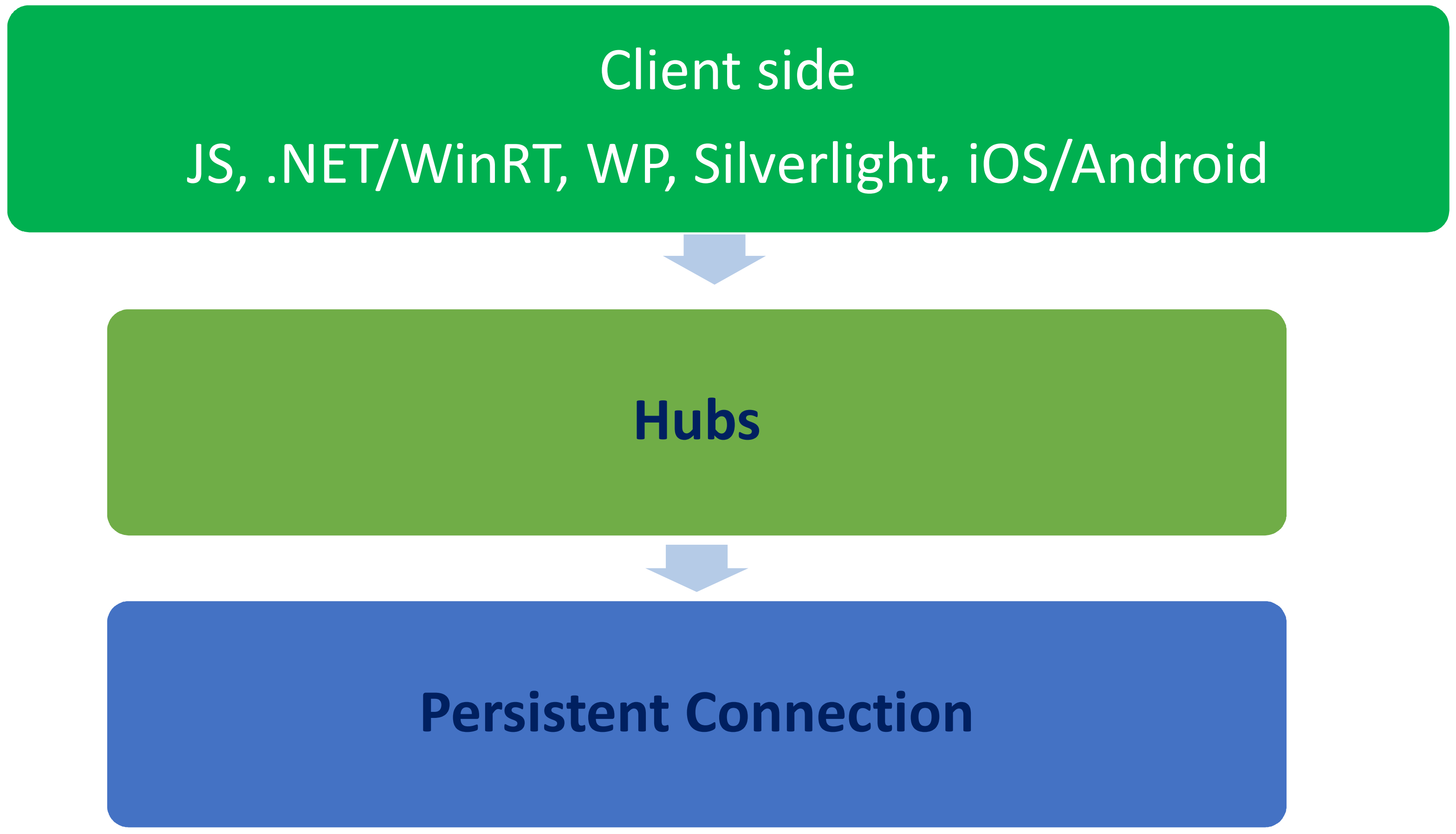
- Apps that require high frequency updates from the server. Examples are gaming, social networks, voting, auction, maps, and GPS apps.
- Dashboards and monitoring apps. Examples include company dashboards, instant sales updates, or travel alerts.
- Collaborative apps. Whiteboard apps and team meeting software are examples of collaborative apps.
- Apps that require notifications. Social networks, email, chat, games, travel alerts, and many other apps use notifications.

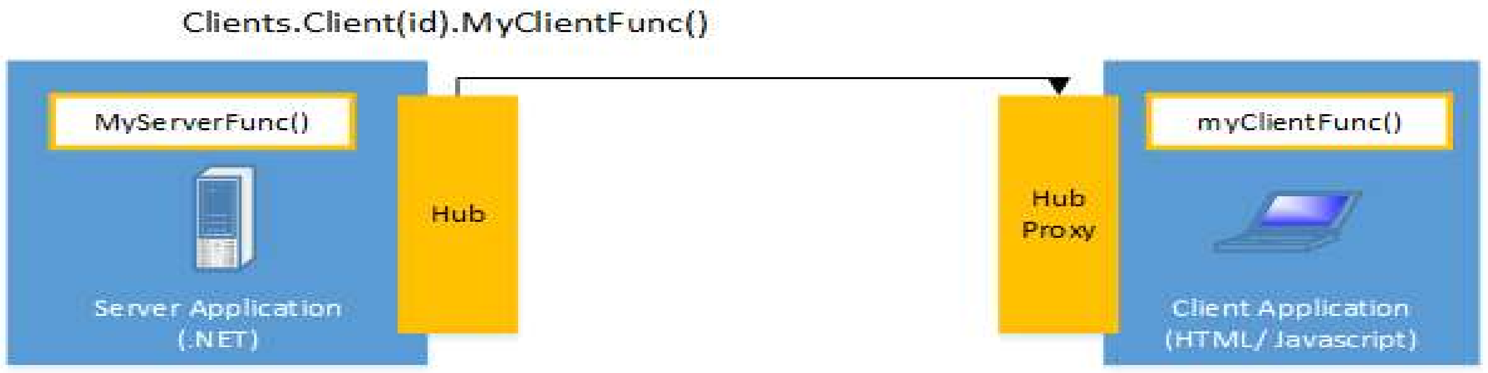
## SignalR

- **SignalR** handles connection management automatically, and lets you broadcast messages to all connected clients simultaneously, like a chat room. You can also send messages to specific clients. The connection between the client and server is persistent, unlike a classic HTTP connection, which is re-established for each communication.

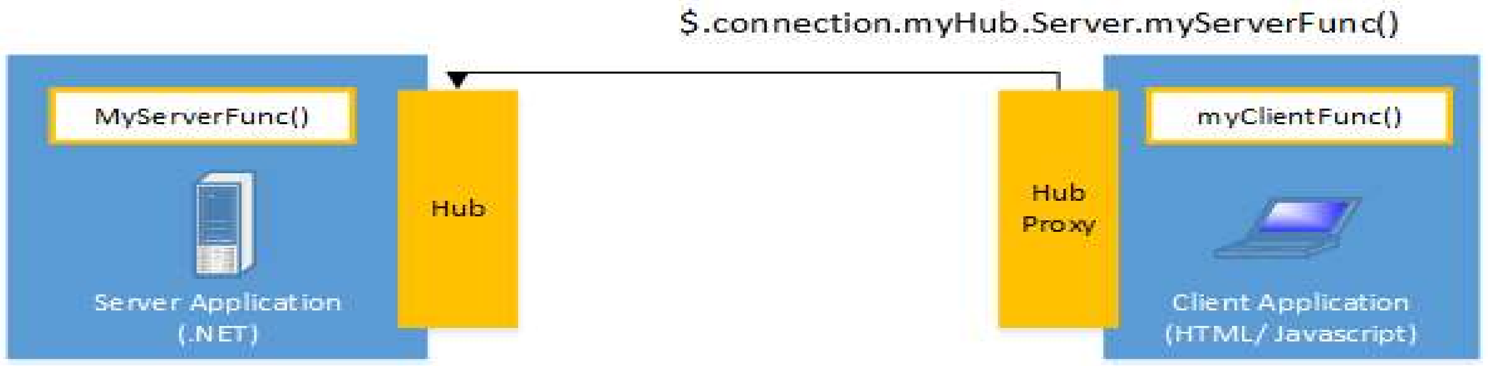
- **SignalR** supports "server push" functionality, in which server code can call out to client code in the browser using Remote Procedure Calls (RPC), rather than the request-response model common on the web today.

## SignalR Connections





Server invocation of client method
MyClientFunc()



Client invocation of server method
MyServerFunc()

## ASP.NET Core SignalR

- **ASP.NET Core SignalR** is an open-source library that simplifies adding real-time web functionality to apps. Real-time web functionality enables server-side code to push content to clients instantly.

- SignalR provides an API for creating server-to-client ***remote procedure calls (RPC).*** The RPCs call JavaScript functions on clients from server-side .NET Core code.

## Transports

**SignalR** supports the following techniques for handling real-time communication (in order of graceful fallback):

- WebSockets
- Server-Sent Events
- Long Polling

**SignalR** automatically chooses the best transport method that is within the capabilities of the server and client.

## Configure SignalR(startup class)

- **ConfigureServices** :

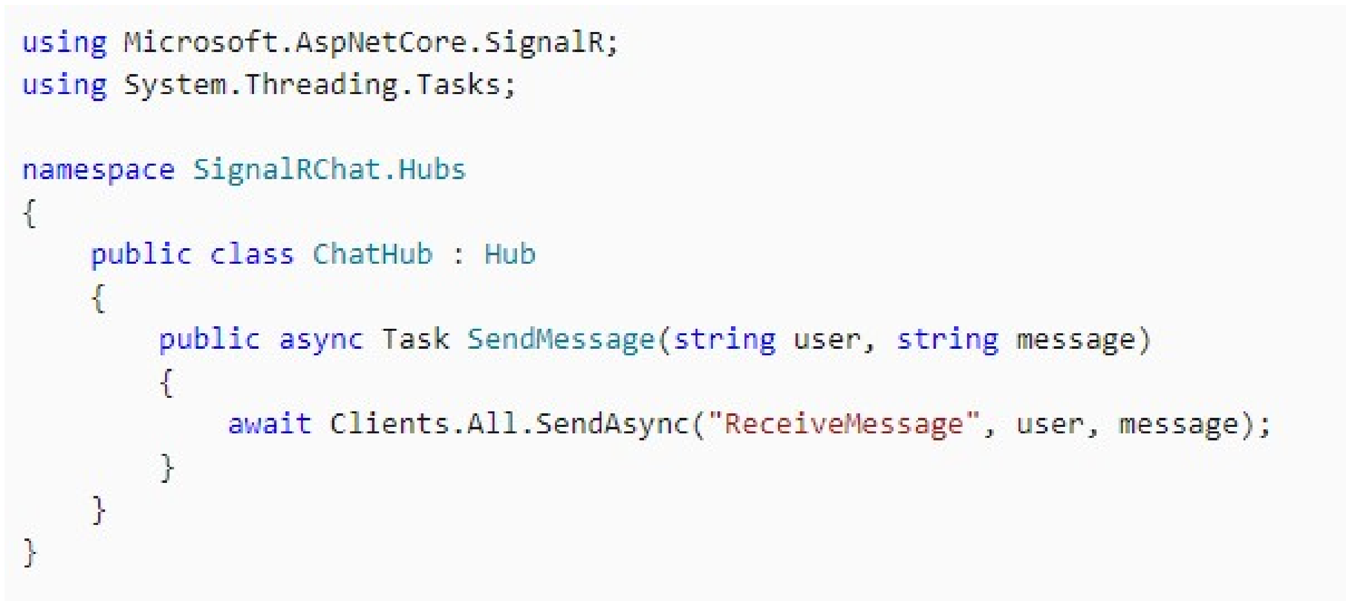
```
services.AddSignalR();
```

- **Configure:**

```
app.MapHub<ChatHub>("/chatHub");
```

## Create a SignalR hub

•In the SignalRChat project folder, create a *Hubs* folder.
•In the *Hubs* folder, create a *ChatHub.cs*



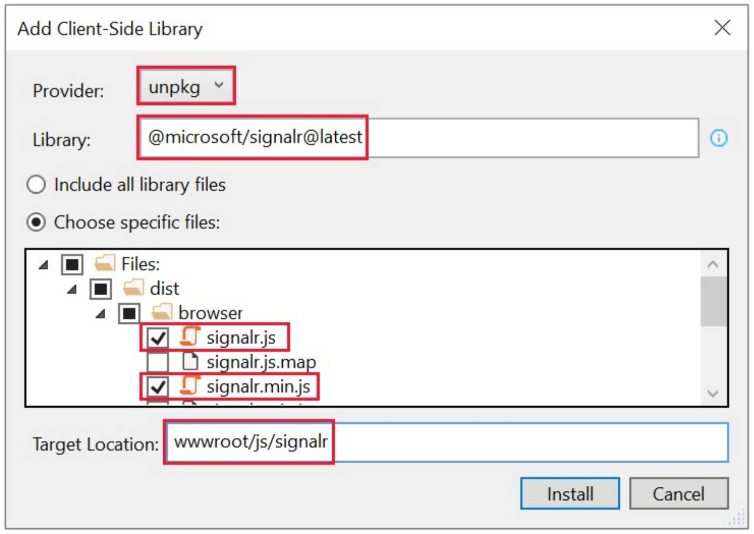
```
using Microsoft.AspNetCore.SignalR;
using System.Threading.Tasks;

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

<script src="https://cdnjs.cloudflare.com/ajax/libs/microsoft-signalr/6.0.1/signalr.js"></script>

## Add the SignalR client library

- In **Solution Explorer**, right-click the project, and select **Add** > **Client-Side Library**.



Provider: unpkg

Library: @microsoft/signalr@latest

Include all library files

Choose specific files:

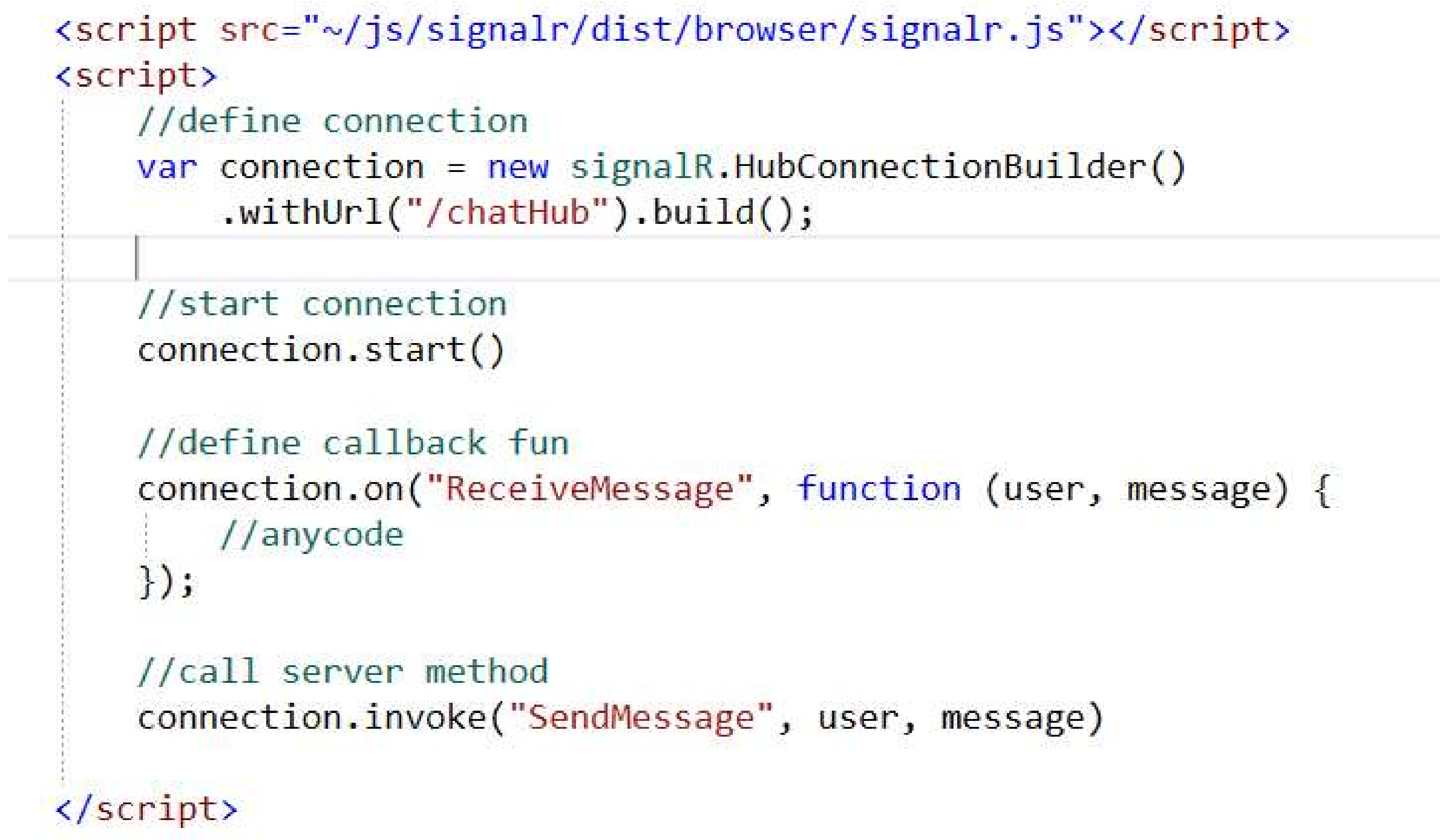
Files:
dist
browser
signalr.js
signalr.js.map
signalr.min.js

Target Location: wwwroot/js/signalr

- **Use a Content Delivery Network (CDN)**

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/microsoft-signalr/6.0.1/signalr.js"></script>
```

## Add SignalR client code



```
<script src="~/js/signalr/dist/browser/signalr.js"></script>
<script>
    //define connection
    var connection = new signalR.HubConnectionBuilder()
        .withUrl("/chatHub").build();

    //start connection
    connection.start()

    //define callback fun
    connection.on("ReceiveMessage", function (user, message) {
        //anycode
    });

    //call server method
    connection.invoke("SendMessage", user, message)

</script>
```