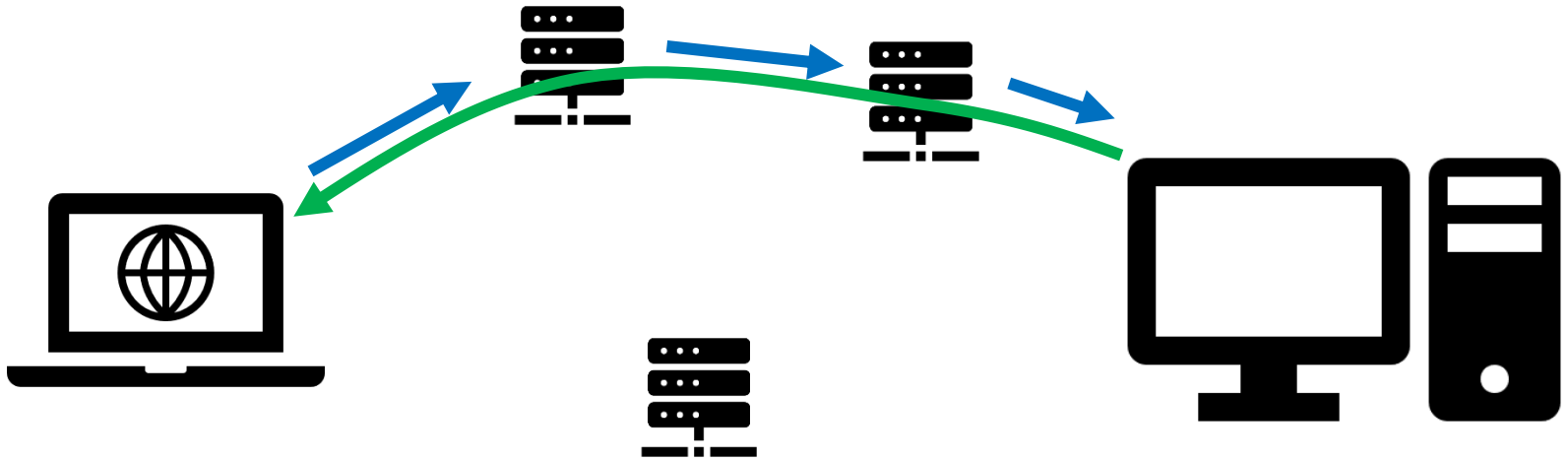
Введение в «SignalR»

Александр Кузнецов

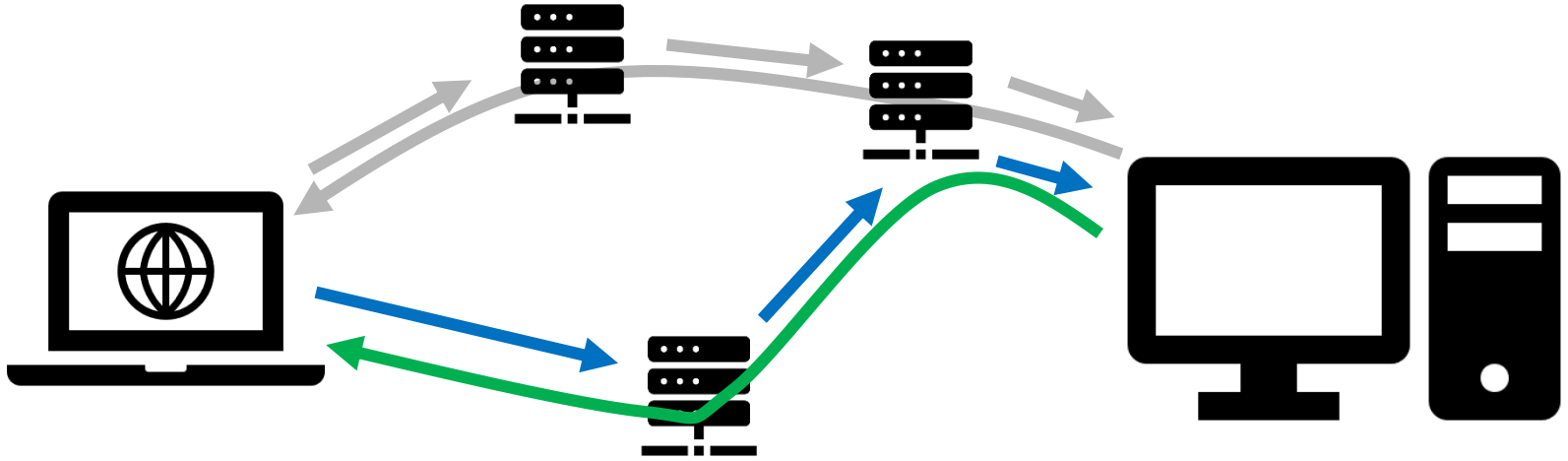
Ведущий программист ROKO Labs

Как работают веб-приложения



Клиент связывается с сервером через цепочку промежуточных узлов и получает ответ

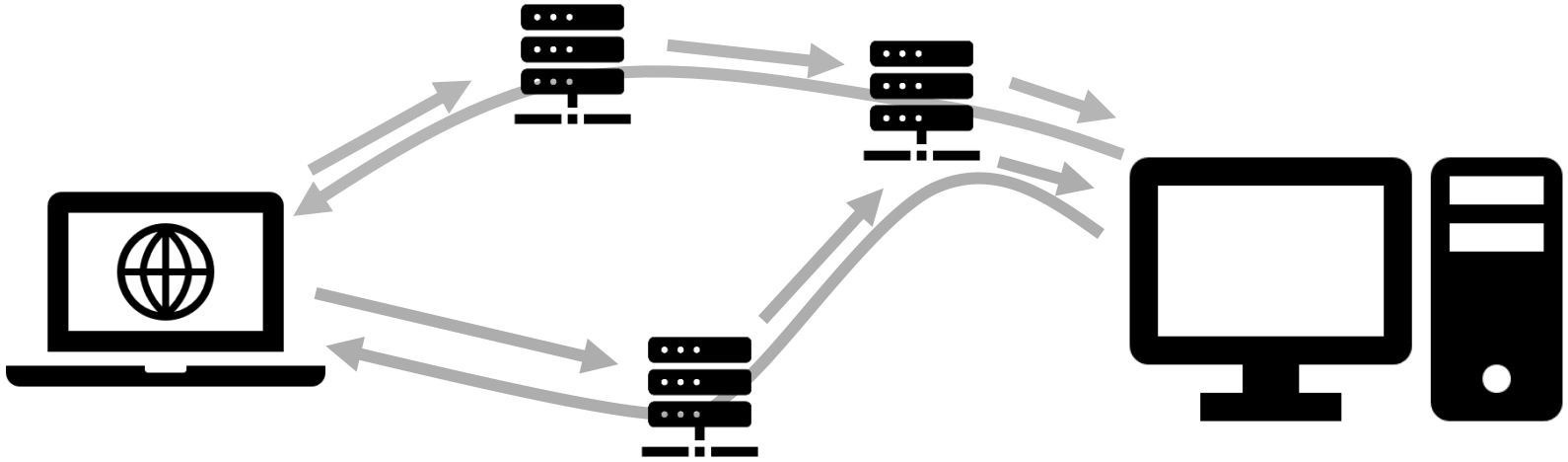
Как работают веб-приложения



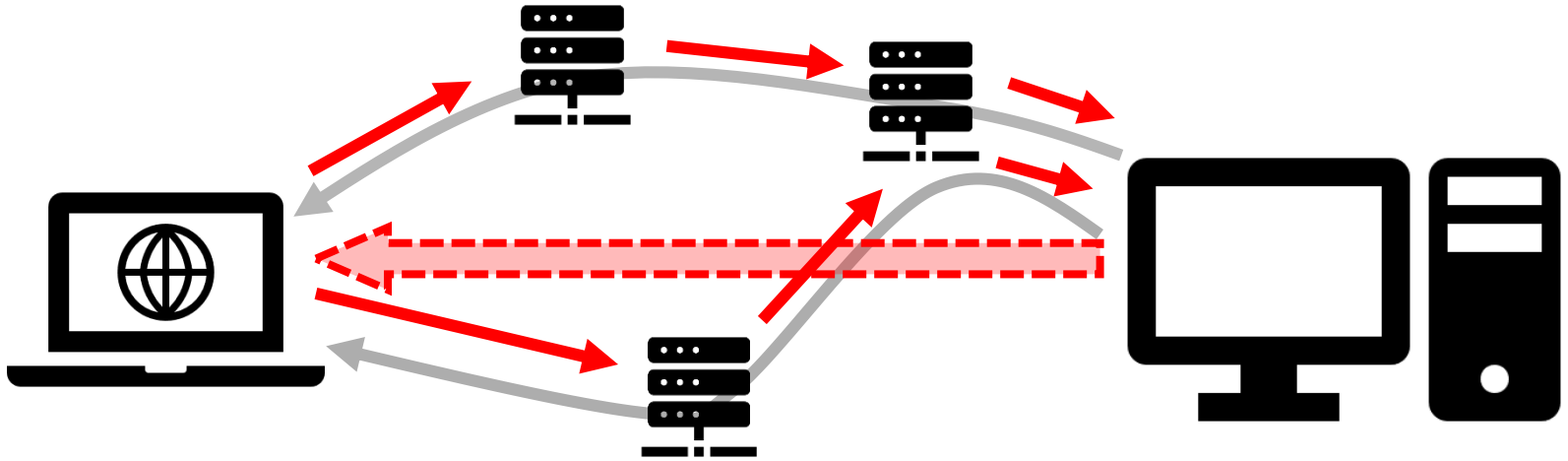
Клиент связывается с сервером через цепочку промежуточных узлов и получает ответ

Новый запрос может пойти через другую цепочку узлов

Как работают веб-приложения



Как работают веб-приложения

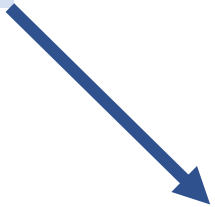


Проблемы:

1. Установка соединения не быстрая операция
2. Сервер не может связаться с клиентом

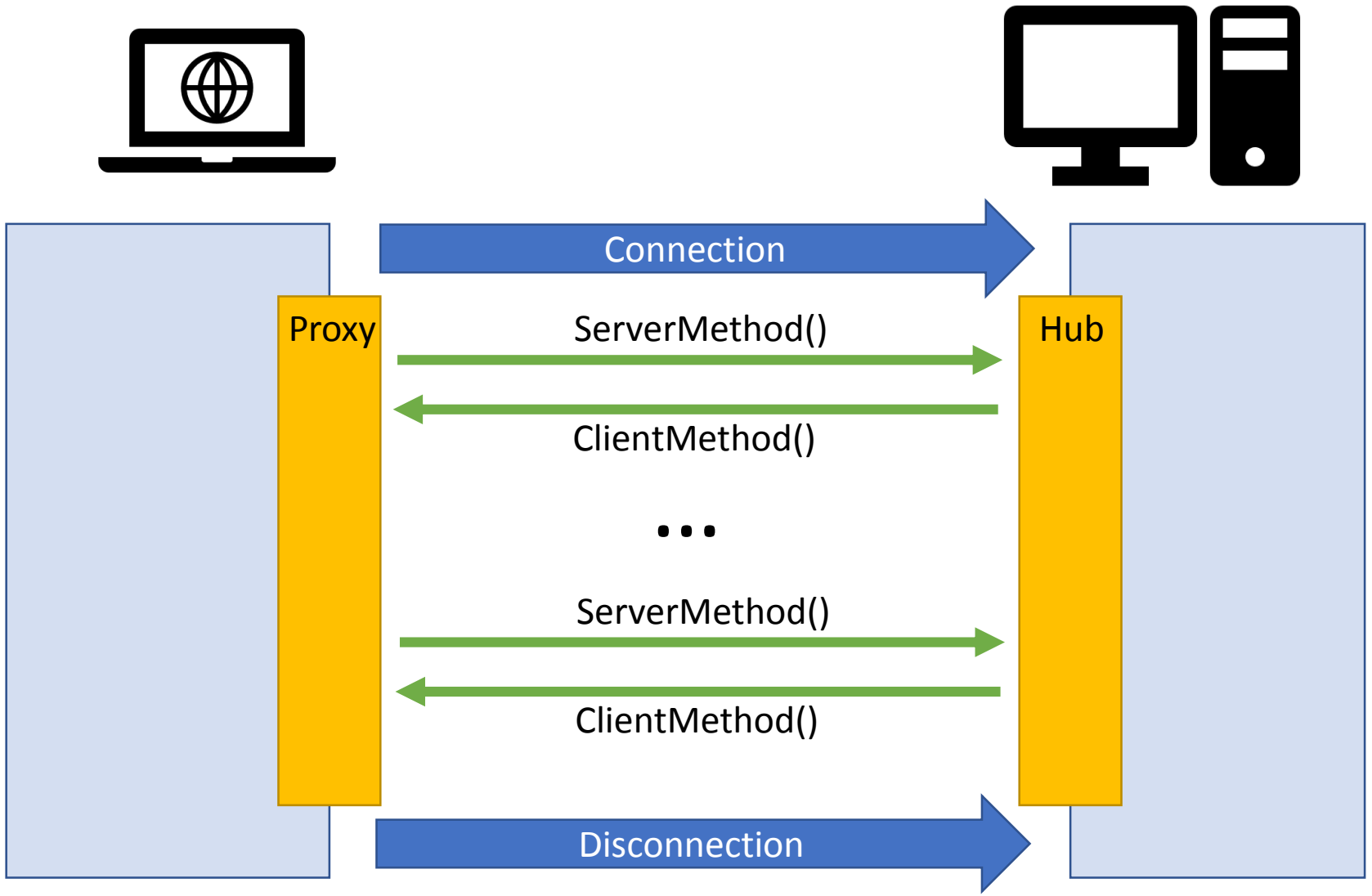
Решения

- Ajax long polling (Comet model)
- Forever Frame (Comet model)
- Server Sent Events (HTML5)
- WebSocket (HTML5)



Microsoft SignalR

Как работает SignalR



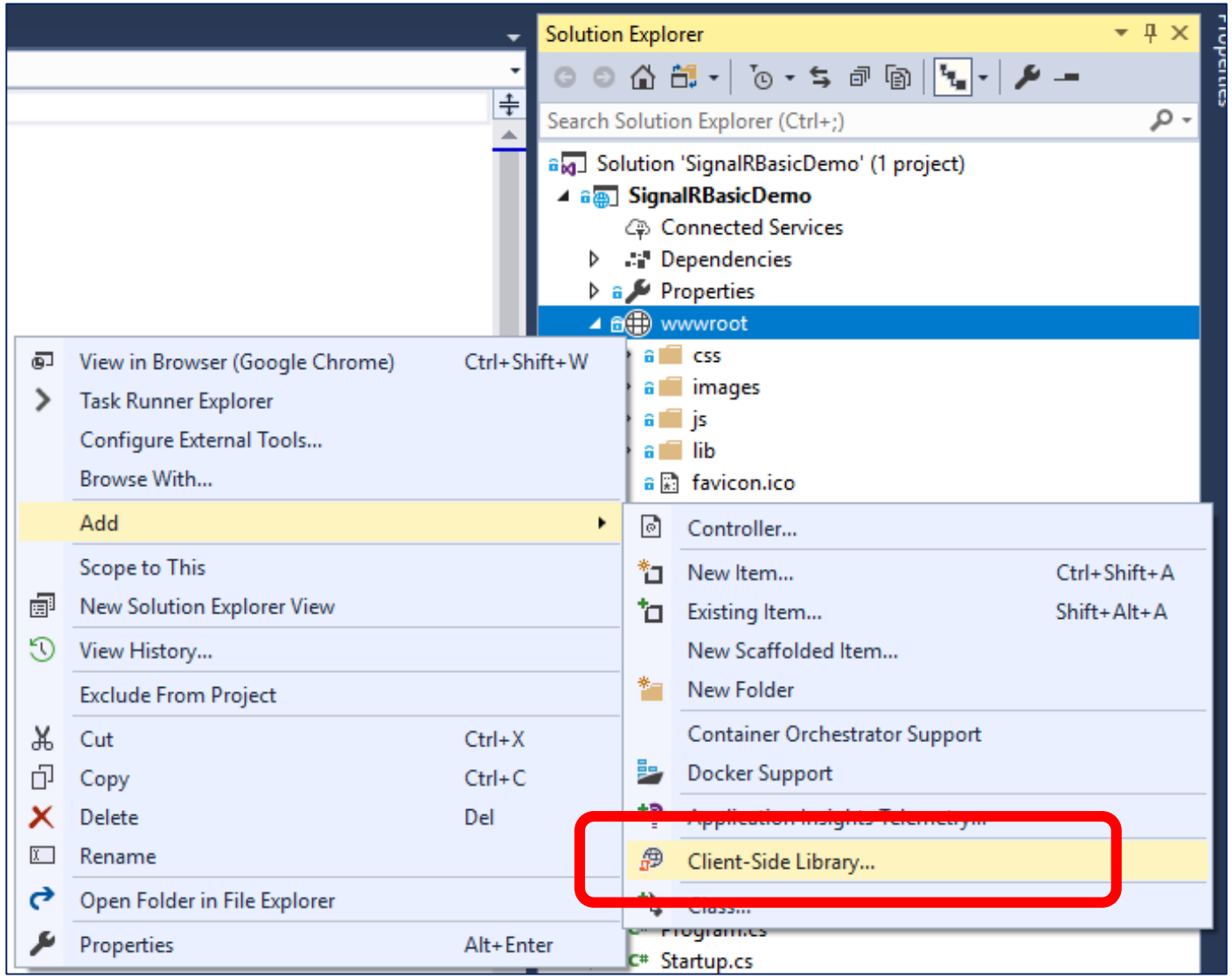
От теории к практике

Версии ПО:

- Visual Studio 2017 15.8 (15.8.6)
- .NET Core SDK 2.1
- SignalR for ASPNetCore (1.0.4)
- (jQuery 3.3.1)

https://docs.microsoft.com/en-us/aspnet/core/tutorials/signalr

От теории к практике

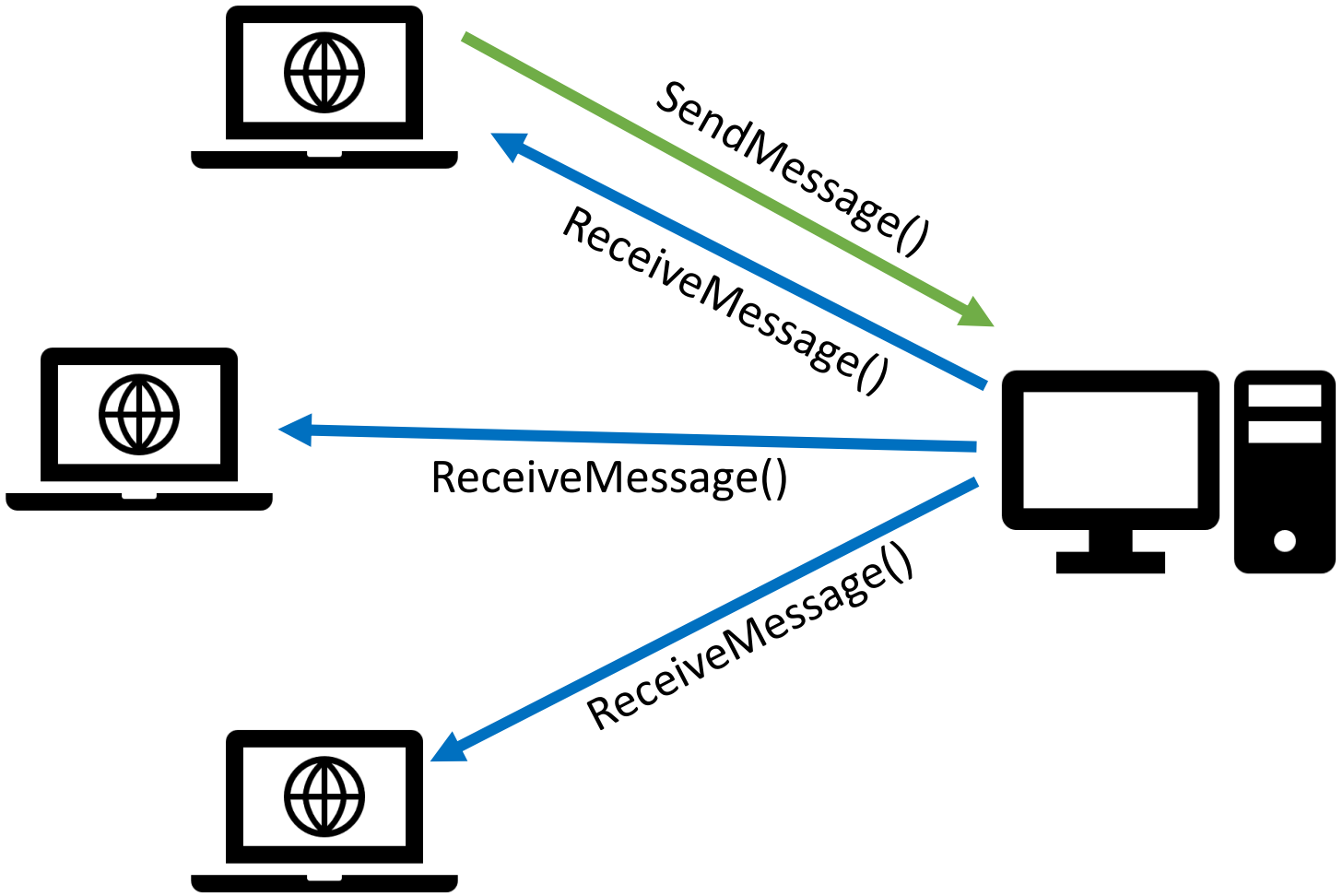


От теории к практике – код хаба

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

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

От теории к практике – логика работы хаба

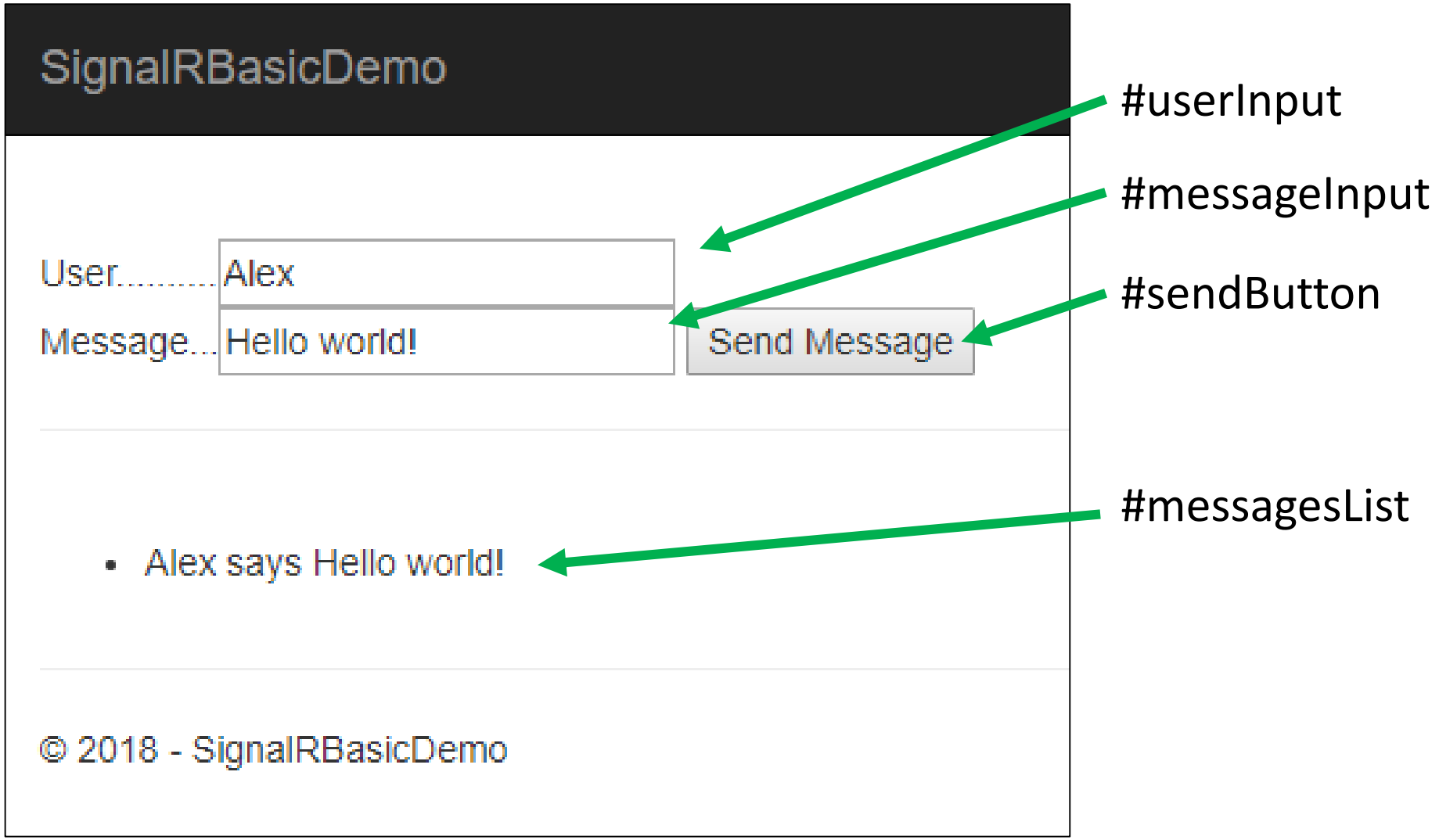


От теории к практике – конфигурация

```
using SignalRBasicDemo.Hubs;

namespace SignalRBasicDemo
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddSignalR();
        }
        public void Configure(IApplicationBuilder app,
                              IHostingEnvironment env)
        {
            app.UseSignalR(routes =>
            {
                routes.MapHub<ChatHub>("/chatHub");
            }
        }
    }
}
```

От теории к практике – клиент



От теории к практике – клиент

```
<script src="~/lib/signalr/dist/browser/signalr.js"></script>

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

connection.on("ReceiveMessage", function (user, message) {
    var msg = user + " says " + message;
    var li = document.createElement("li");
    li.textContent = encodedMsg;
    document.getElementById("messagesList").appendChild(li);
});

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

От теории к практике – клиент

```
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();
});
```

Демонстрация

Простейший чат

https://github.com/aspnet/Docs/tree/master/aspnetcore/tutorials/signalr/sample

https://github.com/akuzn1/BasicSignalRDemo

Синтаксические различия версий

“Классический” SignalR:

```
connection.chathub.start()
    .done(function () {
        console.log("MyHub1 Successfully Started");
    })
    .fail(function () {
        console.log("Error: MyHub1 Not Successfully Started");
    });
```

SignalR Core:

```
connection.start()
    .then(function () {
       console.log("Successfully Started");
    })
    .catch(function (err) {
       console.error(err.toString());
    });
```

Синтаксические различия версий

“Классический” SignalR:

```
connection.chathub.server.sendMessage(name, message);

connection.chathub.client.receiveMessage =
    function(name, message) {
        ...
    }
```

SignalR Core:

```
connection.invoke("SendMessage", user, message)

connection.on("ReceiveMessage", function (user, message) { ... });
```

От теории к практике – усложняем задачу

Космическая игра:

- Корабли летают и подбирают астероиды
- За подбор астероидов идёт опыт. При наборе заданного количества опыта происходит обновление кораблей
- Подобранные астероиды периодически обновляются
- Можно перемещаться по игровым уровням

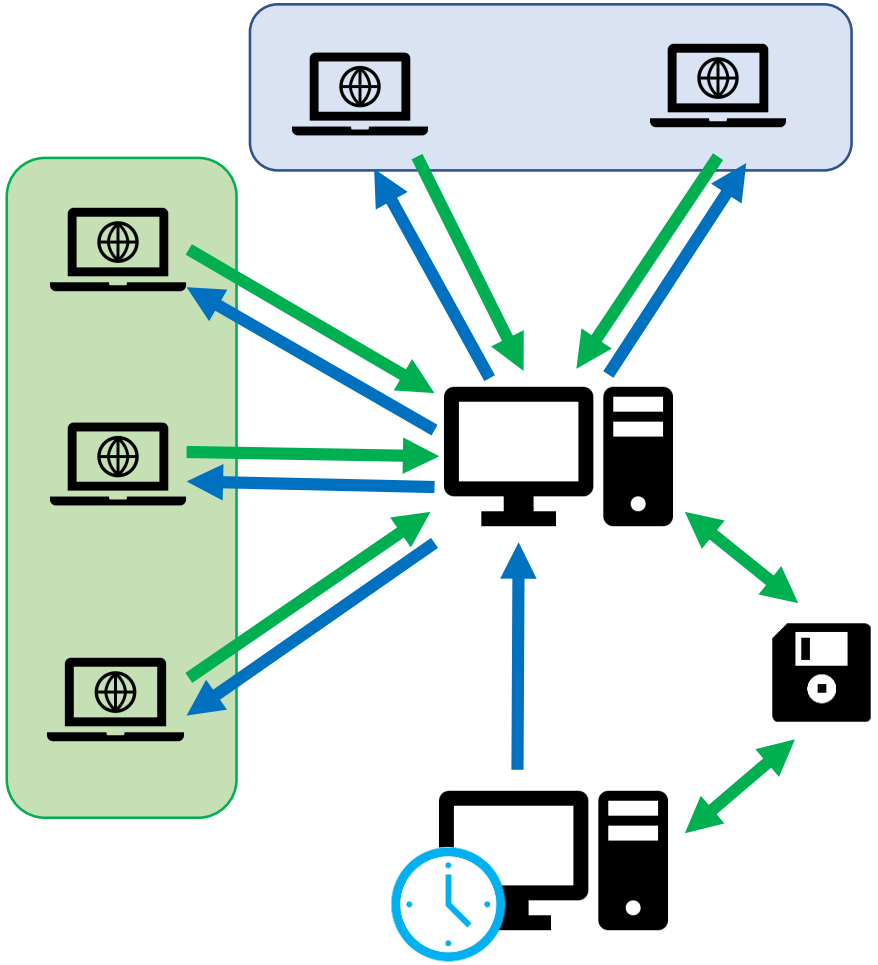
Демонстрация

Простейшая игра

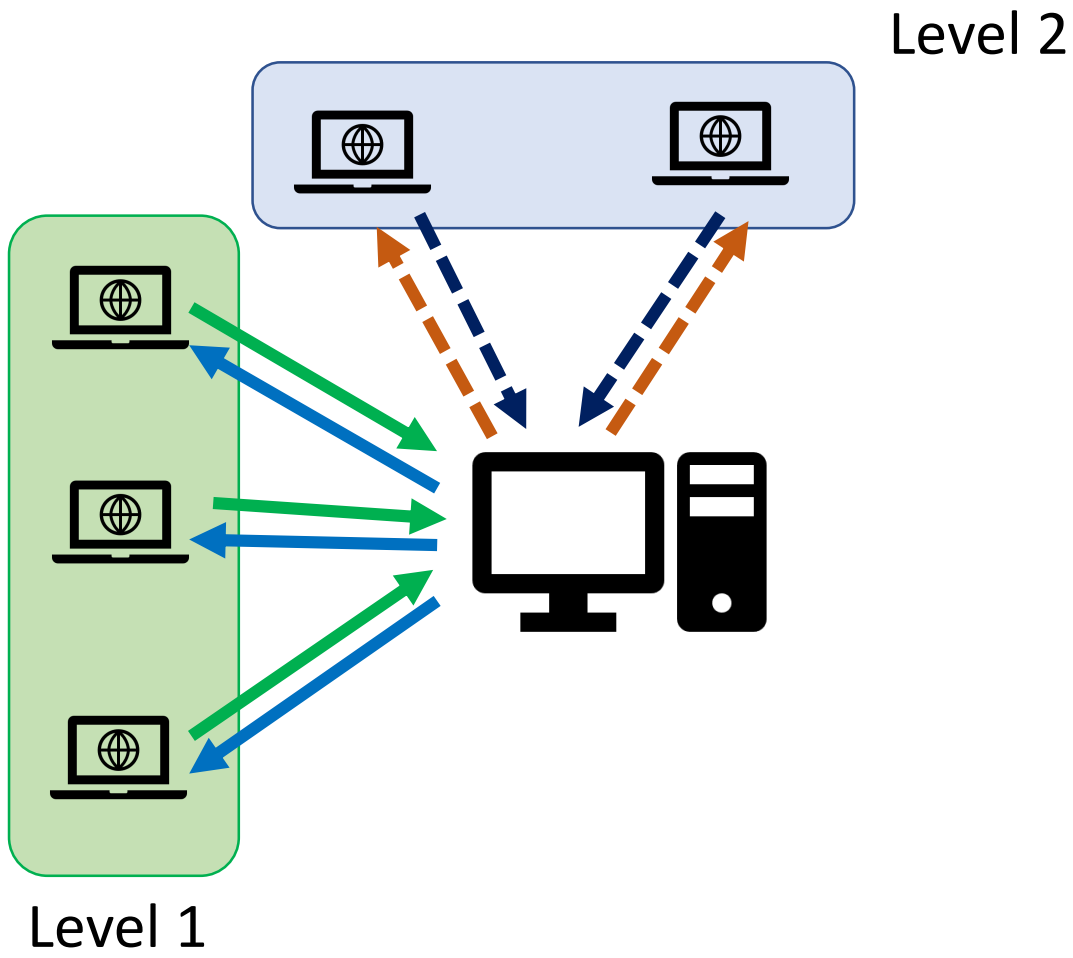
https://github.com/akuzn1/SpaceGameSignalRDemo

Особенности архитектуры

- Клиент только передаёт действия пользователя и показывает результат. Все окончательные расчёты делает сервер
- Начальный объём данных может быть большим
- Обновлением информации и перерасчётами занимается отдельное приложение



Группы



Группы, и не только

Отправить всем:

```
await Clients.All.SendAsync("PlayerAdded", player);
```

Строка подключения:

```
string conStr = Context.ConnectionId;
```

Получатели:

```
All
AllExcept(List<string> exceptedConnectionStrings)
Caller
Others
Clients(List<string> connectionStrings)
Group
Groups
OthersInGroup
User
Users
```

Работа с группами

Добавить в группу:

```
await Groups.AddToGroupAsync(Context.ConnectionId,
        "Level" + player.Level);
```

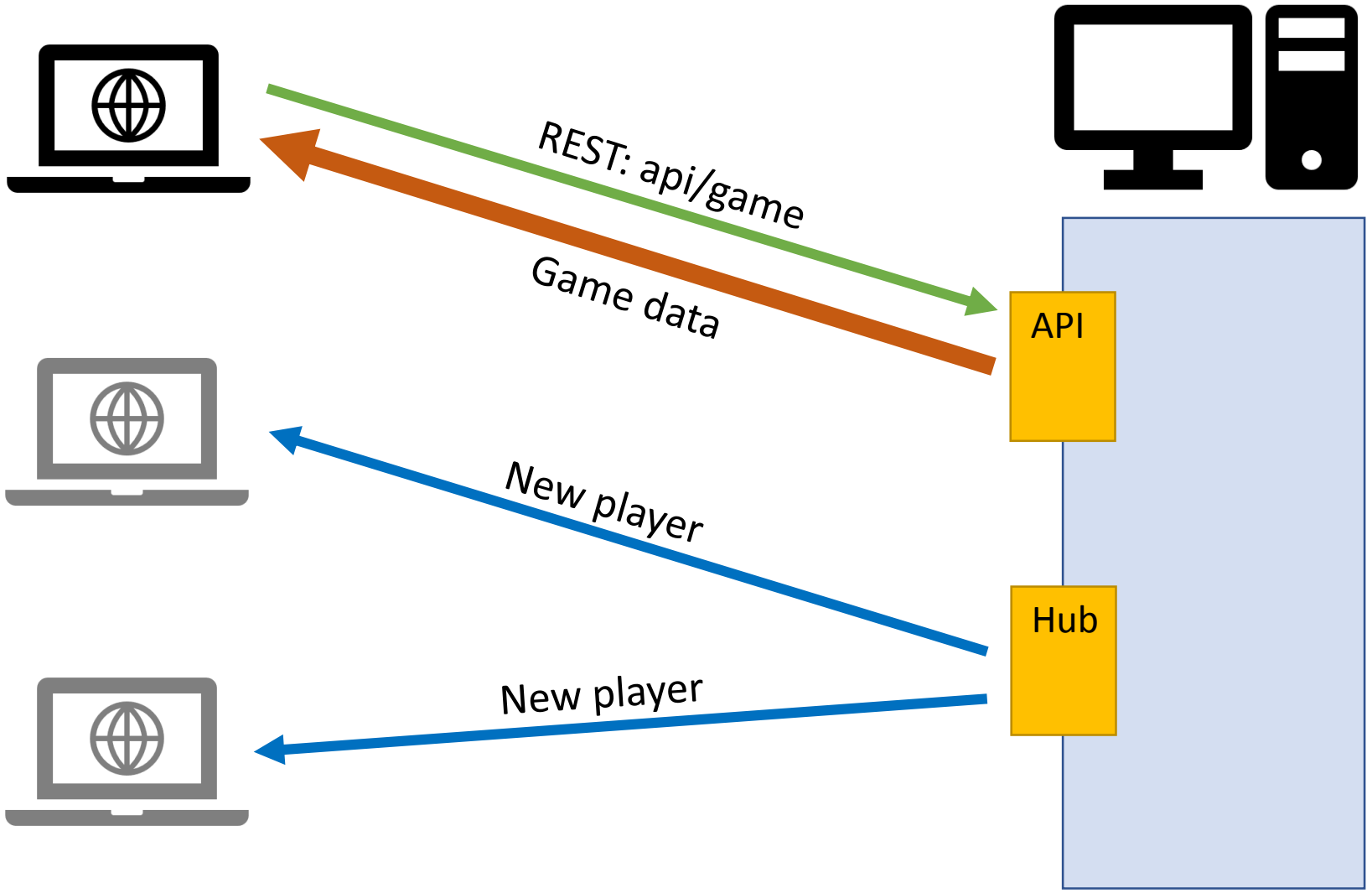
Удалить из группы:

```
await Groups.RemoveFromGroupAsync(Context.ConnectionId,
        "Level" + (player.Level - 1));
```

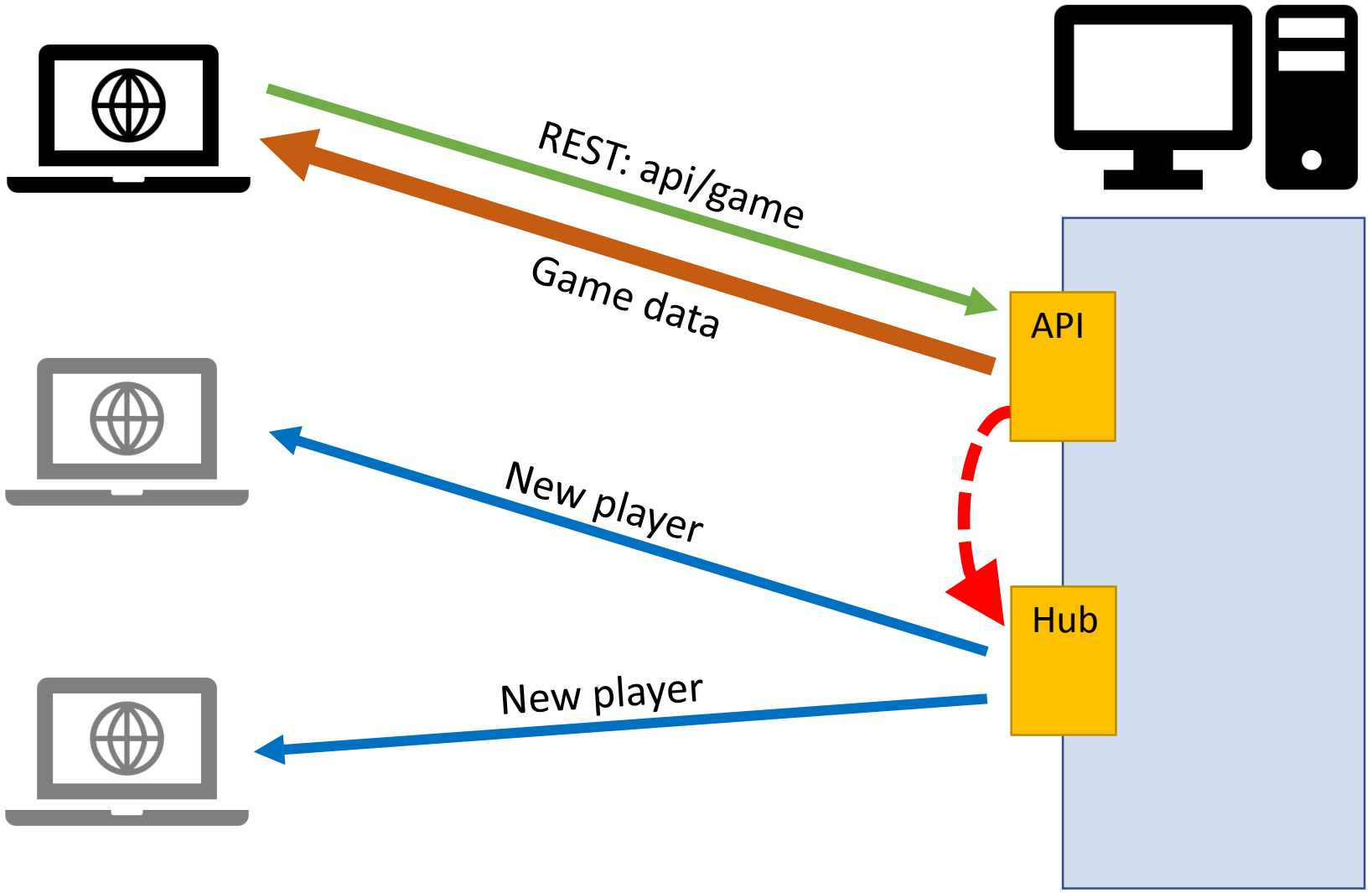
Сообщение на группу:

```
Clients.Group("Level" + player.Level)
        .SendAsync("PlayerAdded", player);
```

Вызов методов хаба из контроллера



Вызов методов хаба из контроллера



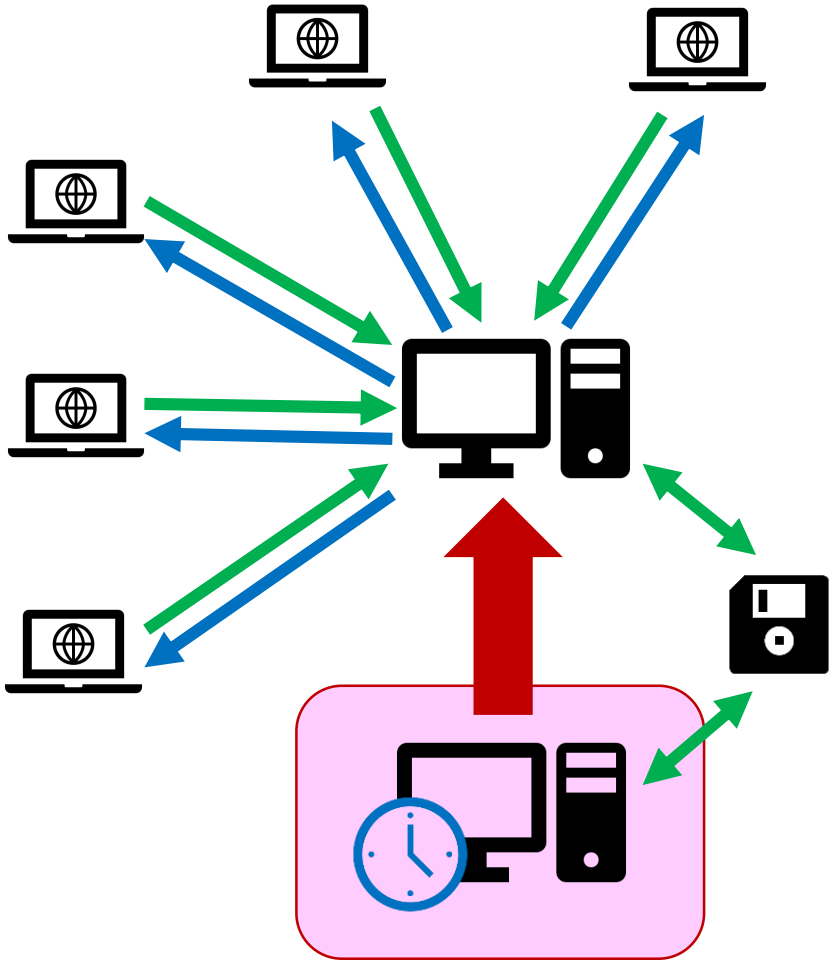
Вызов методов хаба из контроллера

```
public class GameController : ControllerBase
{
    private readonly IHubContext<GameHub> _gameHubContext;

    public GameController(IHubContext<GameHub> gameHubContext)
    {
        _gameHubContext = gameHubContext;
    }

    [HttpPost]
    public ActionResult<GameState> Index([FromBody] string name)
    {
        GameState state = GameLogic.StartGame(name);
        _gameHubContext.Clients.Group("Level" + state.Player.Level)
                .SendAsync("PlayerAdded", state.Player.Ship);
        return state;
    }
}
```

.NET клиент и вызов из другого приложения



.NET клиент и вызов из другого приложения

Устанавливаем клиентский пакет:

```
Install-Package Microsoft.AspNetCore.SignalR -Version 1.0.4
```

Создаём соединение:

```
HubConnection connection;
connection = new HubConnectionBuilder()
    .WithUrl("http://localhost:51582/AdminHub")
    .Build();
```

Подписываемся на события:

```
connection.On<string>("MethodName",
    (p) => { Console.WriteLine(p); });
```

.NET клиент и вызов из другого приложения

Запускаем:

```
connection.StartAsync().Wait();
```

Запускаем с контролем статуса:

```
connection.StartAsync().Wait(settings.MaxConnectionAttemptTimeout);
if (connection.State != ConnectionState.Connected)
    throw new TimeoutException();
```

Используем:

```
connection.InvokeAsync("AddObjects", newObjects);
```

Размеры пакетов

```
ApplicationMaxBufferSize = 32 kb;  // данные от клиента
TransportMaxBufferSize = 32 kb;    // от сервера в буферы и т.д.
```

- По умолчанию: JSON
- Бинарный MessagePack
- Streaming

Протокол MessagePack

Конфигурируем сервер:

```
Install-Package Microsoft.AspNetCore.SignalR.Protocols.MessagePack
```

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSignalR().AddMessagePackProtocol();
}
```

Настраиваем .NET клиент:

```
Install-Package Microsoft.AspNetCore.SignalR.Protocols.MessagePack
```

```
connection = new HubConnectionBuilder()
    .WithUrl("http://localhost:51582/AdminHub")
    .AddMessagePackProtocol()
    .Build();
```

Протокол MessagePack

Настраиваем JS клиент:

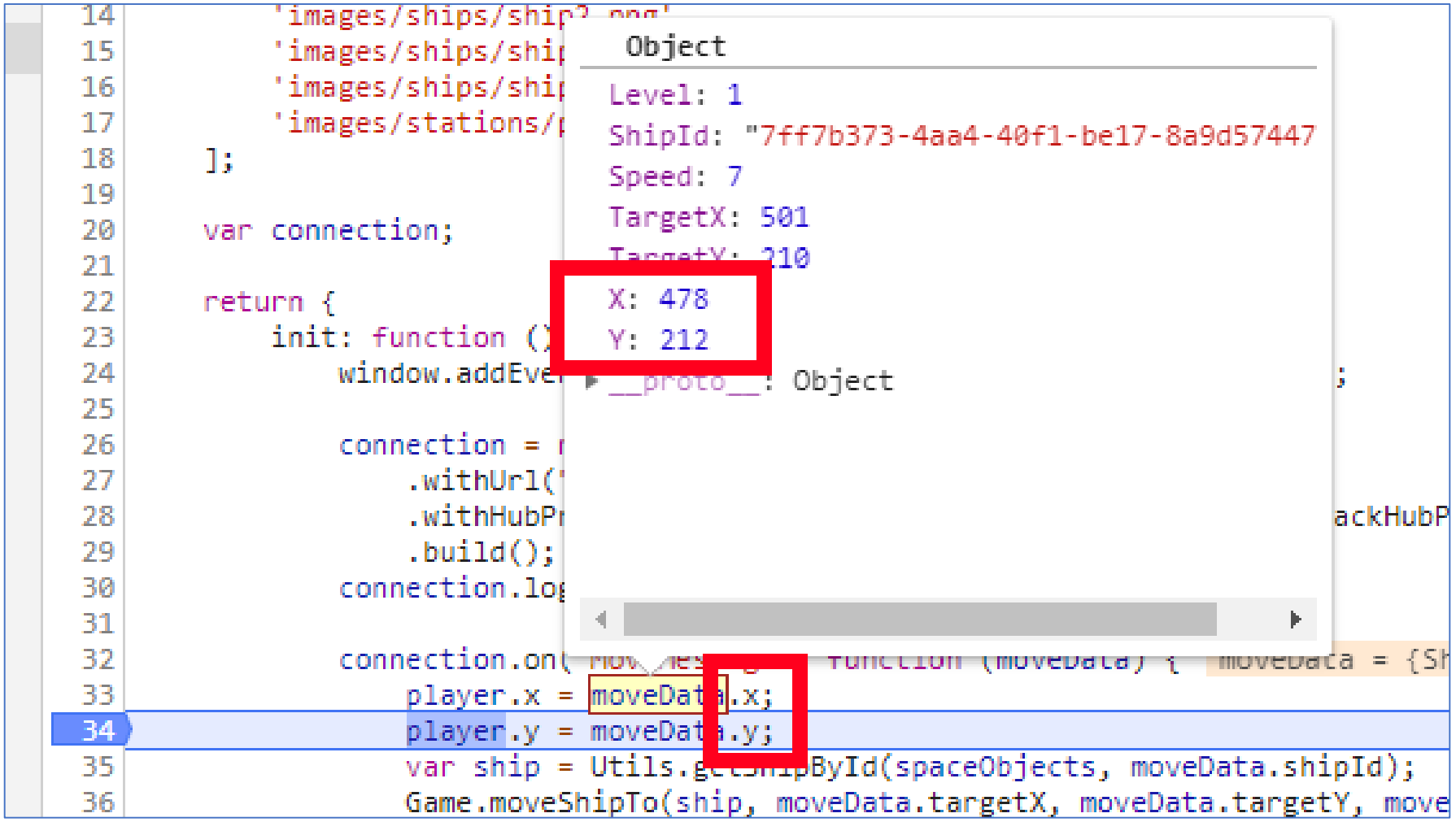
1. `npm install @aspnet/signalr-protocol-msgpack`
2. Руками добавляем библиотеки в папку wwwroot
3. Добавляем ссылки именно в таком порядке

```
<script src="~/lib/signalr/dist/browser/signalr.js"></script>
<script src="~/lib/msgpack5/dist/msgpack5.js"></script>
<script src="~/lib/signalr-protocol-msgpack/
        dist/browser/signalr-protocol-msgpack.js"></script>
```

Настраиваем соединение:

```
connection = new signalR.HubConnectionBuilder()
    .withUrl("/gameHub")
    .withHubProtocol(new signalR.protocols.
                     msgpack.MessagePackHubProtocol())
    .build();
```

Протокол MessagePack



Streaming

```
public class AdminHub : Hub
{
    public ChannelReader<int> Statistic(int maxCount)
    {
        var channel = Channel.CreateUnbounded<int>();
        _ = WriteStatistic(channel.Writer, maxCount);
        return channel.Reader;
    }
    private async Task WriteStatistic(ChannelWriter<int> writer,
            int maxCount)
    {
        for(int i = 0; i < maxCount; i++)
        {
            await writer.WriteAsync(GameLogic.GetActivePlayers());
            await Task.Delay(1000);
        }
        writer.TryComplete();
    }
}
```

Streaming

```
connection.stream("Statistic", 10)
    .subscribe({
        next: (item) => {
            var li = document.createElement("li");
            li.textContent = item;
            document.getElementById("dataList").appendChild(li);
        },
        complete: () => {
            var li = document.createElement("li");
            li.textContent = "Stream completed";
            document.getElementById("dataList").appendChild(li);
        },
        error: (err) => {
            var li = document.createElement("li");
            li.textContent = err;
            document.getElementById("dataList").appendChild(li);
        },
    });
```

Спасибо за внимание!

Александр Кузнецов

Ведущий программист ROKO Labs

akuzn1@gmail.com