WEBSOCKET

by Maksym Khudoliy

softserve

WEBSOCKET

WebSocket is a computer **communication** protocol (over **TCP**) that allows a **persistent bi-directional** network connection between a client (browser) and a server to be opened in real time

Unlike HTTP, in the WebSocket protocol, the browser does not need to constantly ask the server if there are new messages. It is enough to **connect once** to the server and **wait** for the server to send the **message** itself

Examples of tasks where you should consider using WebSocket:

- real time applications
- chat
- online games
- IoT applications
- trading platforms

softserve

WEBSOCKET

To start working with WebSocket, you need to create an object of type **WebSocket**, the parameter of which specifies the URL. URL string must begin with **ws://** (no encryption) or **wss://** (encrypted)

The **interaction** between **client** and **server** is based on an **event system** and **message transmission**:

- to send data to the server, the **send(data)** method is used, the parameters of which are the data to be sent
- the **close()** method is used to close the connection
- to receive data, the "**message**" event is used, into the handler of which an object of the MessageEvent type will be passed, the data property of which contains the transferred data
- the "**error**" event is used to handle errors
- after the connection is established, the "**open**" event is generated
- after the connection is closed, the "**close**" event is generated

softserve

WEBSOCKET

To demonstrate how WebSocket works, we will use the **websocket.org** site, which provides the simplest WebSocket server (wss://echo.websocket.org), it just returns the messages it receives

Let's create the following files:

- **index.js** – main script file
- **index.html** – Web page, only needed to include index.js

WEBSOCKET

index.js file:

```
const s = new WebSocket("wss://echo.websocket.org");
s.addEventListener("open", (e) => {
   console.log("connected");
   s.send("Hello WebSocket!");
   console.log("client -> server: Hello WebSocket!");
});
s.addEventListener("message", (e) => {
   console.log("client <- server:", e.data);
});
```

WEBSOCKET

```
s.addEventListener("close", (e) => {
    console.log("disconnected");
});
setTimeout(() => {
    s.send("Hello World!");
    console.log("client -> server: Hello World!");
}, 1000);
setTimeout(() => s.close(), 2000);
```

WEBSOCKET

We create an object of type WebSocket. Add a handler to the open event so that after the connection is established, immediately send a message to the server. Add a handler to the message event to receive messages from the server. Add a handler for the close event. With a slight delay, we send another message to the server and close the connection

After running the code, the browser console will display the following output:

```
connected
client -> server: Hello WebSocket!
client <- server: Hello WebSocket!
client -> server: Hello World!
client <- server: Hello World!
disconnected
```

softserve


TO
THE
FUTURE
softserve