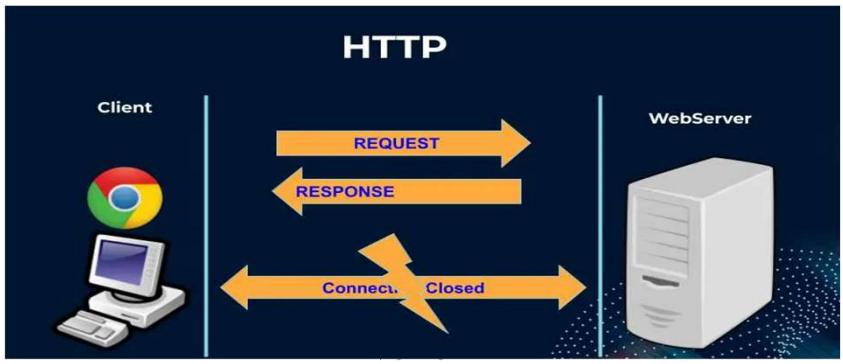
1

1. Introduction to WebSockets

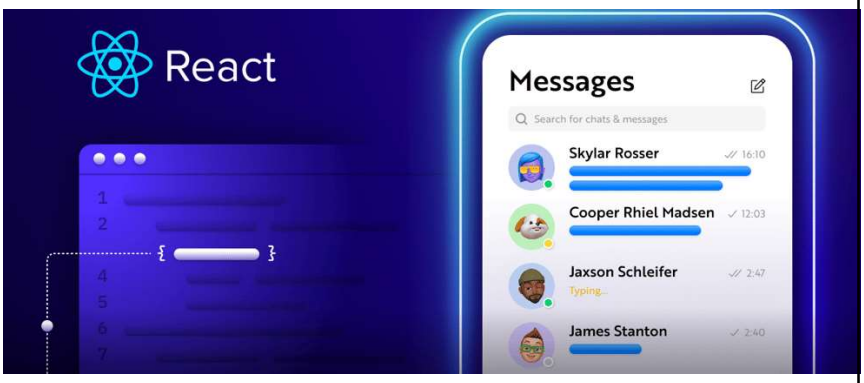
WebSockets are a protocol for full-duplex communication channels over a single, long-lived TCP connection. They allow for real-time data transfer between a client (such as a web browser) and a server with low overheads and low latency.



2

2. Use Cases

- Chat Applications
- Live News Updates
- Online Gaming
- Financial Tickers
- Collaborative Tools



3

3. Advantages of WebSocket

- Real-time Communication
- Low Latency:
- Reduced Overhead
- Scalability
- Efficiency

4

4. Disadvantages of WebSocket

- Complexity
- Compatibility
- Security Concerns
- Firewall and Proxy Issues
- Resource Management

5

5. Server side implementations

- Node.js
  - Socket.IO
  - WebSocket-Node
  - ws
- Java
  - Jetty
- Ruby
  - EventMachine
- Python
  - pywebsocket
  - Tornado
- Erlang
  - Shirasu
- C++
  - libwebsockets
- .NET
  - SuperWebSocket

6

5. Create the Node.js server

1. Create new directory
   mkdir websockets
2. install ws, a WebSockets library for Node.js:
   npm install ws
3. Open the websockets directory using your preferred text editor, like VSCode

5. Create the Node.js server

Open the Package.json file and add the following:

```
{
  "type": "module",
  "dependencies": {
    "ws": "^8.12.0"
  }
}
```

5. Create the Node.js server

Create a index.js file with the content as below

```
import { WebSocketServer } from 'ws';

const wss = new WebSocketServer({ port: 8080 });

wss.on('connection', function connection(ws) {
  ws.on('message', function message(data) {
    console.log('received: %s', data);
  });

  ws.send('something');
});
```


5. Create the Node.js server

Update libraries

npm update

Run server using Node JS

Node index.js

5. Create native React apps using Websocket

Create websocket objects:

```
const webSocket = new WebSocket('ws://my-websocket-server.com');
```

Handling connection:

```
webSocket.onopen = (event) => {
  console.log('WebSocket connection opened:', event);
};
```


5. Create native React apps using Websocket

Handling **incoming** messages:

```
webSocket.onmessage = (event) => {
  console.log('Received from server:', event.data);
};
```

Sending messages to the server

```
webSocket.send('Hello server');
```

5. Create native React apps using Websocket

Handling connection error and closure:

```
webSocket.onerror = (event) => {
  console.log('WebSocket error:', event);
};

webSocket.onclose = (event) => {
  console.log('WebSocket connection closed:', event);
};
```

Closing the WebSocket connection

```
webSocket.close();
```


6. Example

Server:

```
const WebSocket = require('ws');

const server = new WebSocket.Server({ port: 8080 });

server.on('connection', ws => {
  console.log('Client connected');

  ws.on('message', message => {
    console.log(`Received: ${message}`);
    // Gửi lại thông điệp tới client
    ws.send(`Server received: ${message}`);
  });

  ws.on('close', () => {
    console.log('Client disconnected');
  });
});
console.log('WebSocket server is running on ws://localhost:8080');
```

6. Example

Client

```
import { useEffect, useState } from 'react';
import { View, Text, Button } from 'react-native';
export default function TestSocket() {
    const [data, setData] = useState('');
    const webSocket = new WebSocket('ws://10.0.2.2:8080');
    webSocket.onopen = function () {
        alert('Connect successful');
    }
    webSocket.onerror = function (error) {
        console.log('Error: ' + JSON.stringify(error));
    }
    webSocket.onmessage = function (e) {
        console.log('data response', e.data);
    }
```


15

6. Example

Client

```
function SendData() {
    var data = {
        title: 'Tieu de',
        message: 'Hãy nhận lấy thông báo của tôi',
        picture: 'abc.jpg'
    }
    webSocket.send(JSON.stringify(data));
}
function SendArrayData() {
        const array = new Float32Array(5);
        for (var i = 0; i < array.length; ++i) {
            array[i] = i / 2;
        }
        webSocket.send(array);
}
```


16

6. Example

Client

```
return (
        <View><Text>abcd</Text>
            <Button onPress={SendData} title='SendData'> </Button>
            <Button onPress={SendArrayData} title='Send array data'> </Button></View>

    );
}
```


17

18