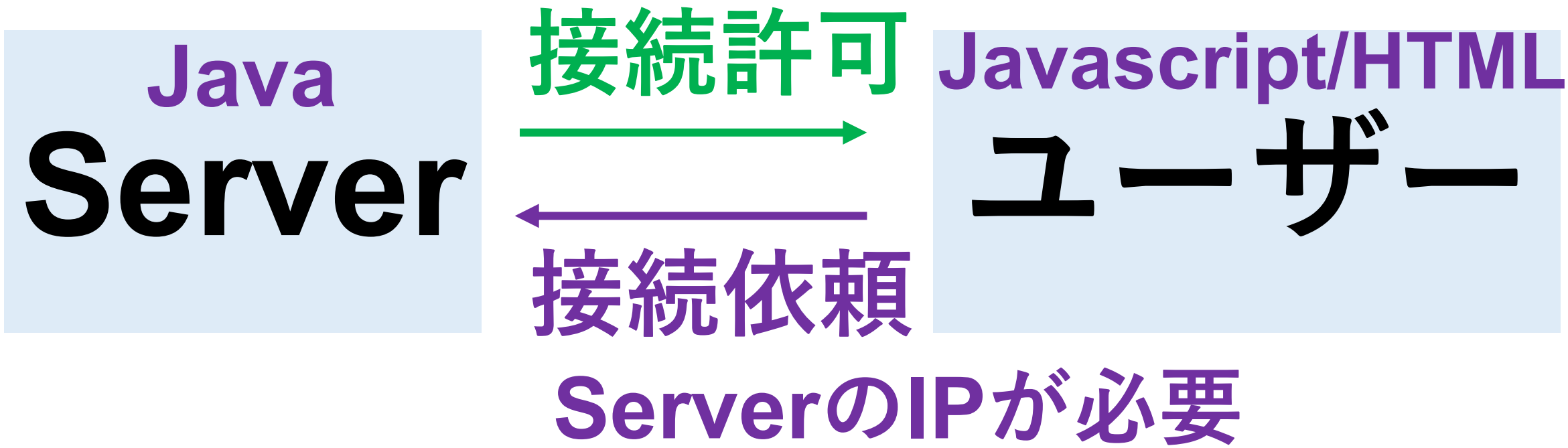
Websockets: JavaとJavascript/HTML連携

Savong Bou
Center for Computational Sciences
University of Tsukuba

Websocket

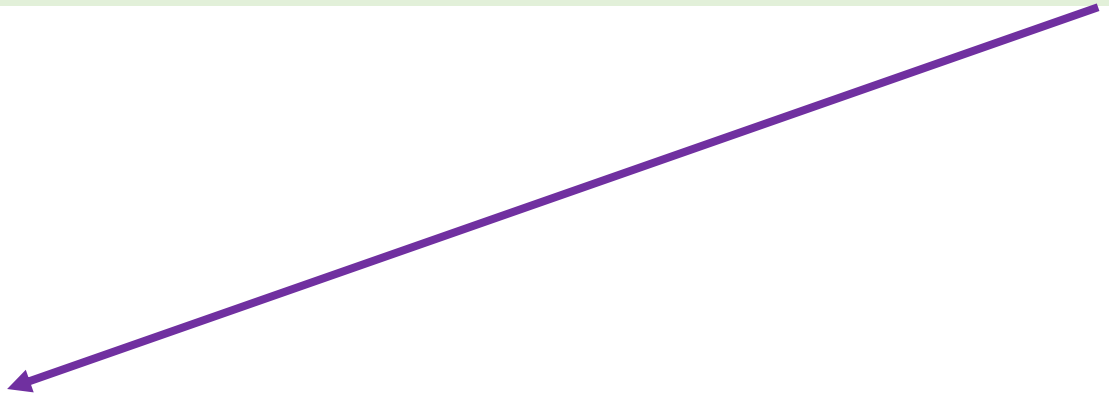
- サーバー側とユーザー側を常時接続状態にしておいて
  双方向通信ができるようにする技術になります。



Javascript

Javascript: Websocket使い方

```
var connection = new WebSocket(【通信を行うURL】);
```



Javascript

URL:「ws」&「wss」

- 一般的なURLは「http://」もしくは「https://」から始まりますが、WebSocketの場合は特別な通信を行うため
「ws://」もしくは「wss://」から始まるURLになります。

- 「ws://」と「wss://」の違いは「http:// https://」の違いと同じ

例

ServerのIP
か
同じマシン場合「localhost」

ポート

```
var connection = new WebSocket(ws://127.0.0.1:4444);
```

Javascript

イベント処理とメソッド

```
var connection = new WebSocket(【通信を行うURL】);

//通信が接続された場合
connection.onopen = function(e) { };

//エラーが発生した場合
connection.onerror = function(error) { };

//メッセージを受け取った場合
connection.onmessage = function(e) { };

//データを送信するメソッド
connection.send();

//通信が切断された場合
connection.onclose = function() { };
```

Javascript

テキストデータを受信する方法

```
connection.onmessage = function(e) {

    //受信したメッセージをポップアップ表示
    alert(e.data);

};
```

Java

Java: Websocket使い方

```
new InetSocketAddress(TCP_PORT)
```

ソケット・アドレスを作成します。

この場合、IPアドレスはワイルドカード・アドレスで、
ポート番号は指定された値です。

ポート

Java

Java: Websocket使い方

ソケット・アドレスを作成方法

コンストラクタ

コンストラクタ	説明
`InetSocketAddress(int port)`	ソケット・アドレスを作成します。この場合、IPアドレスはワイルドカード・アドレスで、ポート番号は指定された値です。
`InetSocketAddress(String hostname, int port)`	ホスト名とポート番号からソケット・アドレスを作成します。
`InetSocketAddress(InetAddress addr, int port)`	IPアドレスとポート番号からソケット・アドレスを作成します。

Java

WebsocketServer作成・開始

```
public class WebsocketServer extends WebSocketServer {

    private static int TCP_PORT = ….;

    private static Set<WebSocket> conns;

    public WebsocketServer() {
        super(new InetSocketAddress(TCP_PORT));
        conns = new HashSet<>();
    }

……
}
```

Java

イベント処理とメソッド(1/2)

```
//通信が接続された場合
@Override
    public void onOpen(WebSocket conn, ClientHandshake handshake) {
        conns.add(conn);
        conn.send("hello!!");
}


//通信が切断された場合
    @Override
    public void onClose(WebSocket conn, int code, String reason, boolean remote) {
        conns.remove(conn);
}
```

Java

イベント処理とメソッド(2/2)

```
//メッセージを受け取った場合
@Override
    public void onMessage(WebSocket conn, String message) {
        System.out.println("Message from client: " + message);
        for (WebSocket sock : conns) {
            sock.send("SENDING BACK" + message);
        }
    }

//エラーが発生した場合
    @Override
    public void onError(WebSocket conn, Exception ex) {
if (conn != null) {
            conns.remove(conn);
}
}
```

Java

サーバーを起動

メインメソッドMain()の中に
WebsocketServerを呼び

```
new WebsocketServer().start();
```