
WebSocket Tutorial (with. Socket.IO)

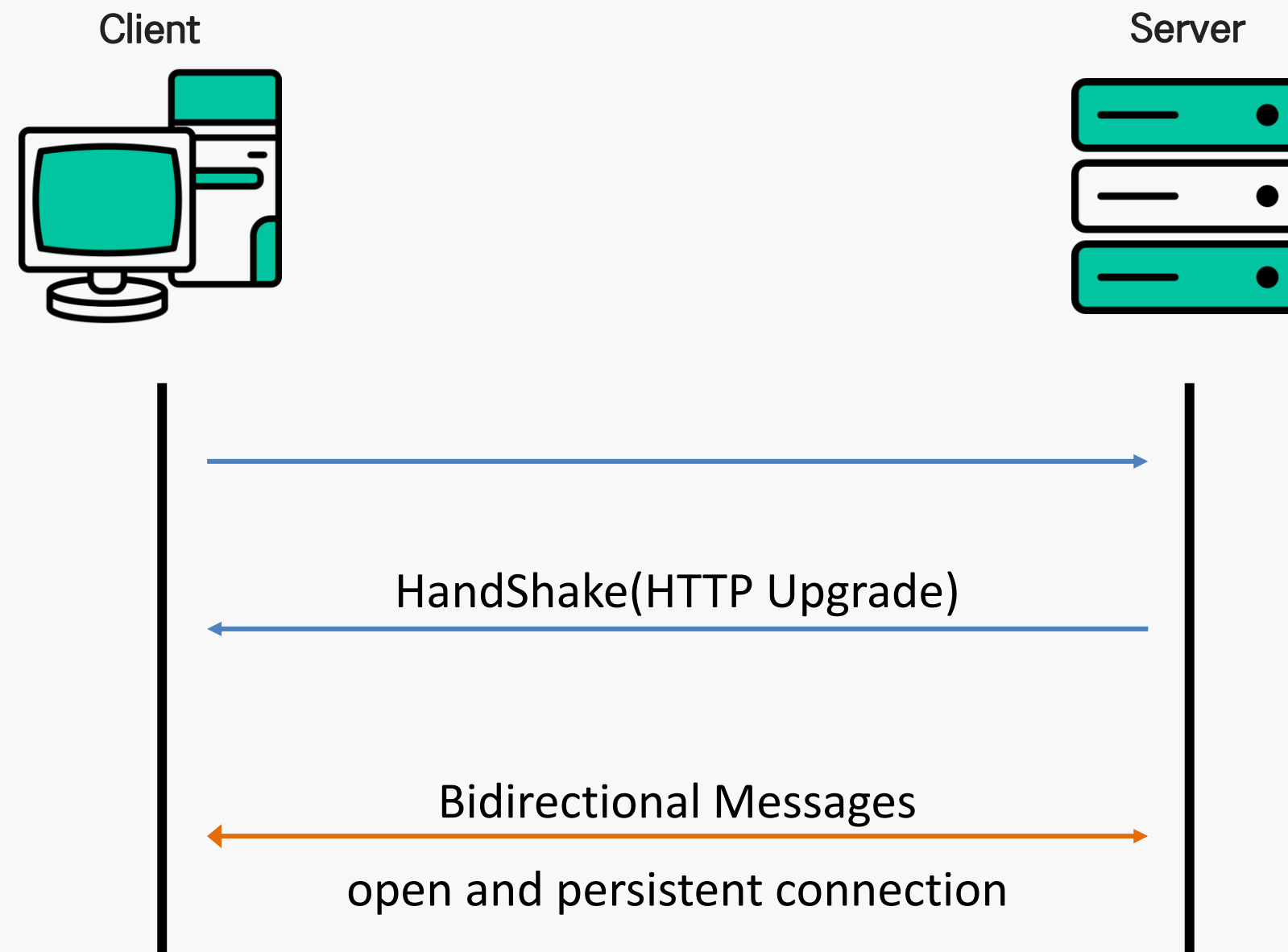
Table of Contents

- I 웹소켓(WebSocket)이란?
- II 웹소켓 사용 예제 (with. node.js)
- III Socket.IO 소개
- IV Socket.IO 설치 및 설정 (with. Express)
- V Socket.IO 이벤트 송수신
- VI Socket.IO 사용 예제 (with. Express)

웹소켓(WebSocket)

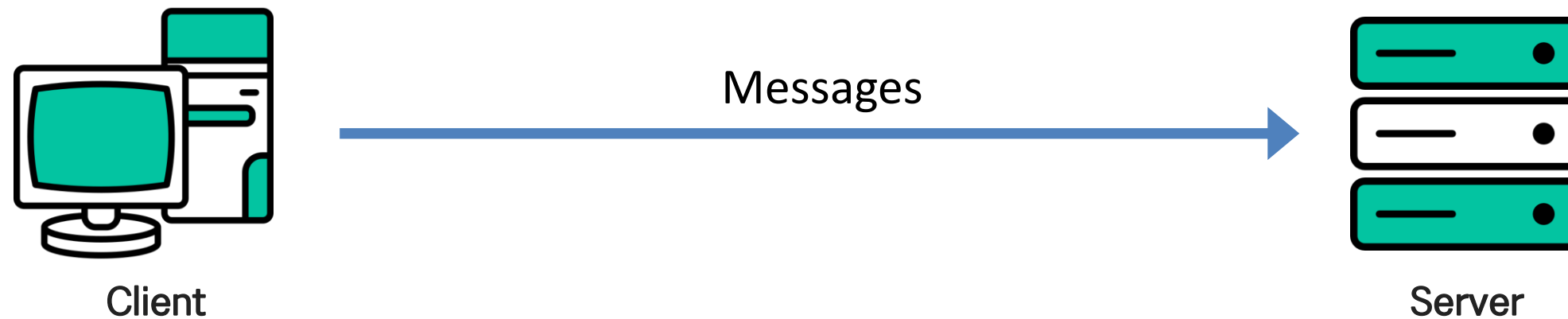
웹소켓(WebSocket)

클라이언트와 서버(브라우저와 서버)를 연결하고 실시간으로 통신이 가능하도록 하는 첨단 통신 프로토콜 하나의 TCP 접속에 전이중(duplex) 통신 채널을 제공



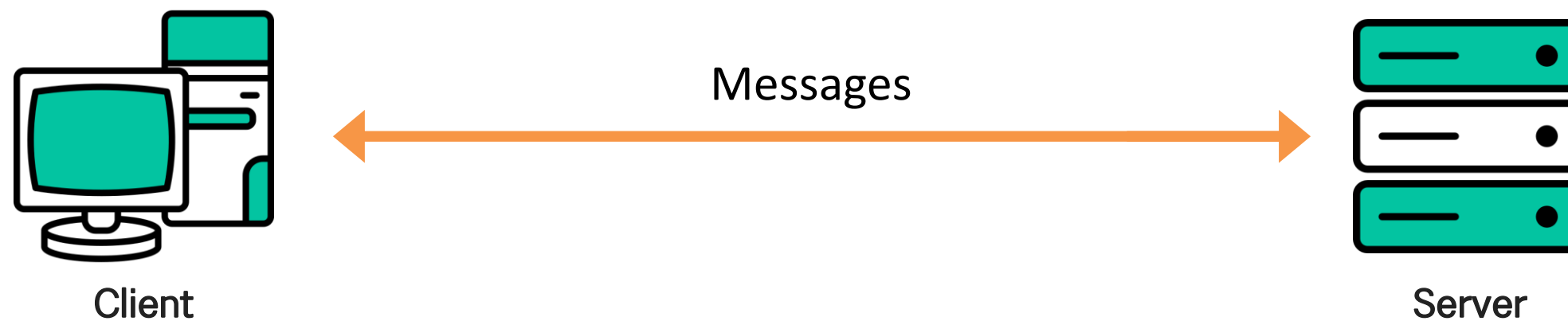
HTTP - 단방향 통신

클라이언트에서 서버로 Request를 보내면, 서버는 클라이언트로 Response를 보내는 방식으로 동작한다.
* 무상태(Stateless)로 상태를 저장하지 않는다.



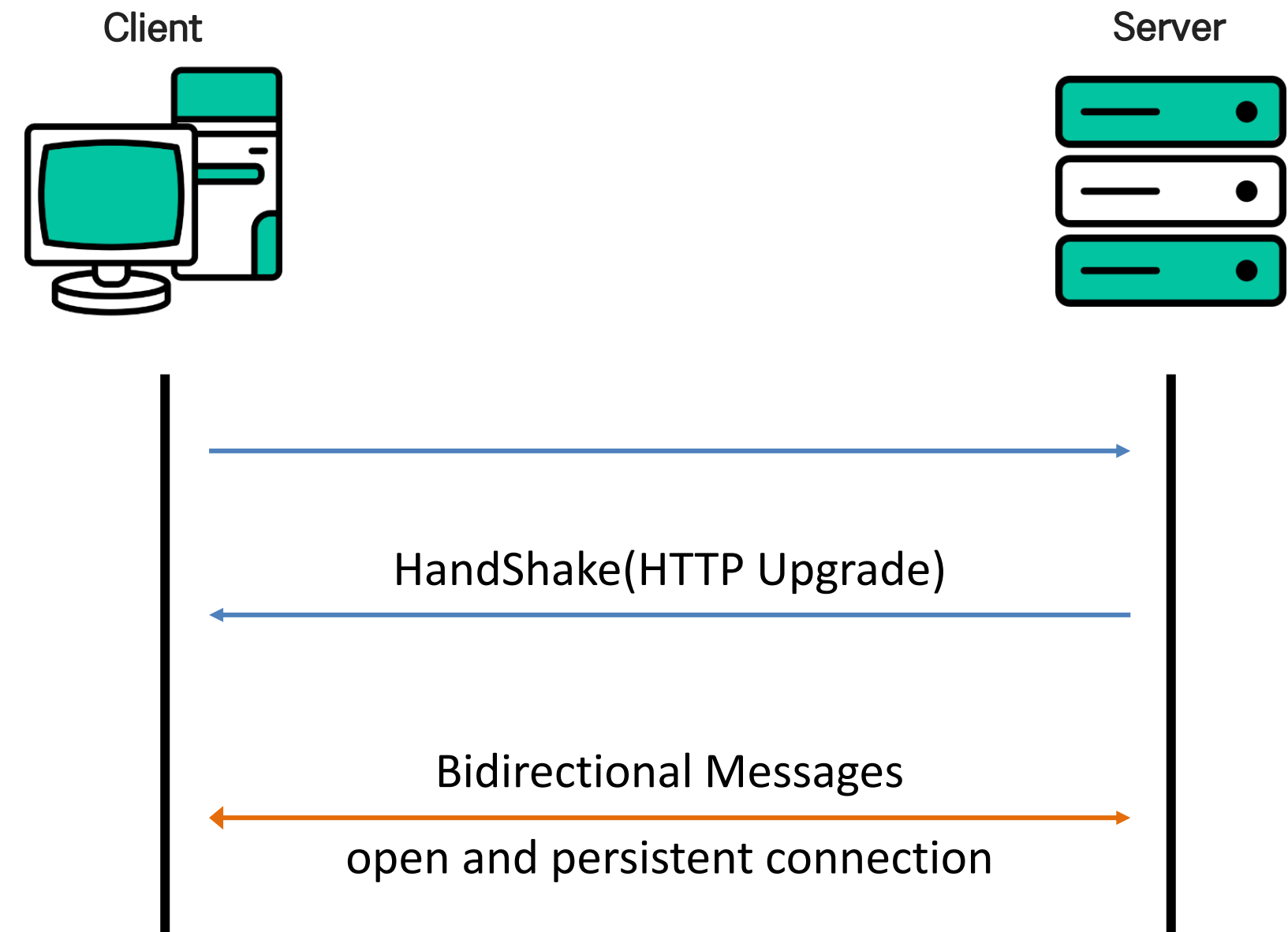
웹소켓(WebSocket) - 양방향 통신

연결이 이루어지면 클라이언트가 요청하지 않아도 데이터가 저절로 서버로부터 올 수 있다.
따라서 HTTP처럼 별도의 요청을 보내지 않아도 데이터를 수신할 수 있다.



웹소켓 동작원리

1. TCP연결 처럼 핸드셰이크를 이용해 연결을 맺는다.
2. HTTP 업그레이드 헤더를 사용하여 HTTP 프로토콜에서 웹소켓 프로토콜로 변경한다.
즉, 최초 접속시에는 HTTP 프로토콜을 이용해 핸드셰이킹을 한다.
3. 연결이 맺어지면 어느 한쪽이 연결을 끊지 않는 이상 영구적인 동일한 채널이 맺어지고, HTTP 프로토콜이 웹소켓 프로토콜로 변경된다.



* HTTP 포트 80, HTTPS 포트 443 위에서 동작

II 웹소켓 사용 예제(with. node.js)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>WebSocket Example</title>
</head>
<body>
<script>
  // 웹소켓 전역 객체 생성
  var ws = new WebSocket("ws://localhost:3000");

  // 연결이 수립되면 서버에 메시지를 전송한다
  ws.onopen = function(event) {
    ws.send("Client message: Hi!");
  }

  // 서버로 부터 메시지를 수신한다
  ws.onmessage = function(event) {
    console.log("Server message: ", event.data);
  }

  // error event handler
  ws.onerror = function(event) {
    console.log("Server error message: ", event.data);
  }
</script>
</body>
</html>
```

index.html

```
var WebSocketServer = require("ws").Server;
var wss = new WebSocketServer({ port: 3000 });

// 연결이 수립되면 클라이언트에 메시지를 전송하고 클라이언트로부터의 메시지를 수신한다
wss.on("connection", function(ws) {
  ws.send("Hello! I am a server.");
  ws.on("message", function(message) {
    console.log("Received: %s", message);
  });
});
```

server.js

Node.js 디렉터리 설치

* npm 설치 필요

```
$ mkdir [디렉터리 이름] && cd [디렉터리 이름]
$ npm init --yes
$ npm install ws --save
```

디렉터리 설치 후

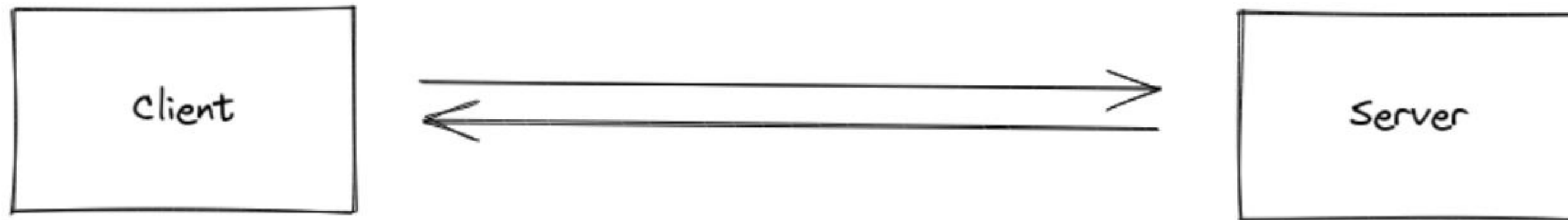
1. 디렉터리에 index.html 및 server.js 붙여넣기
2. "\$ node server" 명령어 실행
3. index.html 파일 실행

Socket.IO

Socket.IO란?

브라우저와 서버 간의 실시간, 양방향, 그리고 이벤트 기반 통신을 가능하게 해주는 라이브러리로, WebSocket 프로토콜 위에 구축되어 있다.

즉, **Socket** 통신을 위한 모듈이라 볼 수 있다.




- Socket.IO 공식홈페이지(www.socket.io)

Socket.IO 설치 (socket.io와 express 설치)

```
$ mkdir [디렉터리 이름] && cd [디렉터리 이름]  
$ npm init --yes  
$ npm install --save --save-exact socket.io express
```

* npm 설치필요

Socket.IO 이벤트 통신 방법



```
//이벤트 수신
socket.on('받을 이벤트명' , (msg) => {});

//이벤트 송신
socket.emit('전송할 이벤트명', msg);
```

-> 해당되는 이벤트를 받고, 콜백함수를 실행

-> 이벤트명을 정하고, 메시지를 보냄

Socket.IO 메소드

소켓 메시지 수신

```
// 접속된 모든 클라이언트에게 메시지를 전송한다
io.emit('event_name', msg);

// 메시지를 전송한 클라이언트에게만 메시지를 전송한다
socket.emit('event_name', msg);

// 메시지를 전송한 클라이언트를 제외한 모든 클라이언트에게 메시지를 전송한다
socket.broadcast.emit('event_name', msg);

// 특정 클라이언트에게만 메시지를 전송한다
io.to(id).emit('event_name', data);
```

소켓 메시지 송신

```
// 클라이언트와 소켓IO 연결 여부에 따른 이벤트 실행
io.on('connection/disconnection', (socket) => {});

// 클라이언트에서 지정한 이벤트가 emit되면 수신 발생
socket.on('event_name', (data) => {});
```

Socket.IO 사용 예제

Server측 이벤트 송수신

server.js

```
var app = require('express')();
var server = require('http').createServer(app);

// http server를 socket.io server로 upgrade
var io = require('socket.io')(server);

// localhost:3000으로 서버에 접속하면 클라이언트로 index.html을 전송
app.get('/', function(req, res) {
  res.sendFile(__dirname + '/index.html');
});

server.listen(3000, function() {
  console.log('Socket IO server listening on port 3000');
});

// connection event handler
// connection이 수립되면 event handler function의 인자로 socket이 들어옴
io.on('connection', function(socket) {

});
```

Express를 이용하여 Http 서버를 생성한뒤, 이 서버를 socket.io serve로 upgrade한다

클라이언트가 서버에 접속 시, connection 이벤트가 발생하는데, 이때 connection event handler를 정의한다

connection Event 콜백함 수 예

```
// connection event handler
io.on('connection', function(socket) {

  // 접속한 클라이언트의 정보가 수신되면
  socket.on('login', function(data) {
    console.log('Client logged-in:\n name:' + data.name + '\n userid: ' + data.userid);

    // socket에 클라이언트 정보를 저장한다
    socket.name = data.name;
    socket.userid = data.userid;

    // 접속된 모든 클라이언트에게 메시지를 전송한다
    io.emit('login', data.name );
  });
```

```
// 클라이언트로부터의 메시지가 수신되면
socket.on('chat', function(data) {
  console.log('Message from %s: %s', socket.name, data.msg);

  var msg = {
    from: {
      name: socket.name,
      userid: socket.userid
    },
    msg: data.msg
  };

  // 메시지를 전송한 클라이언트를 제외한 모든 클라이언트에게 메시지를 전송한다
  socket.broadcast.emit('chat', msg);
});

// force client disconnect from server
socket.on('forceDisconnect', function() {
  socket.disconnect();
})

socket.on('disconnect', function() {
  console.log('user disconnected: ' + socket.name);
});
});
```


Client측 이벤트 송수신

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Socket.io Chat Example</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>
  <div class="container">
    <h3>Socket.io Chat Example</h3>
    <form class="form-inline">
      <div class="form-group">
        <label for="msgForm">Message: </label>
        <input type="text" class="form-control" id="msgForm">
      </div>
      <button type="submit" class="btn btn-primary">Send</button>
    </form>
    <div id="chatLogs"></div>
  </div>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
  <script src="/socket.io/socket.io.js"></script>
```

로그인 정보를 보내기 위한 임의의 form

socket.io를 사용하기 위해 “/socket.io/socket.io.js” 를
추가해, socket.io.js 라이브러리를 추가한다.

VI Socket.io 사용 예제(with. Express)

```
<script>
$(function(){
  // socket.io 서버에 접속한다
  var socket = io();

  // 서버로 자신의 정보를 전송한다.
  socket.emit("login", {
    // name: "ungmo2",
    name: makeRandomName(),
    userid: "ungmo2@gmail.com"
  });

  // 서버로부터의 메시지가 수신되면
  socket.on("login", function(data) {
    $("#chatLogs").append("<div><strong>" + data + "</strong> has joined</div>");
  });

  // 서버로부터의 메시지가 수신되면
  socket.on("chat", function(data) {
    $("#chatLogs").append("<div>" + data.msg + " : from <strong>" + data.from.name + "</strong></div>");
  });

  // Send 버튼이 클릭되면
  $("form").submit(function(e) {
    e.preventDefault();
    var $msgForm = $("#msgForm");

    // 서버로 메시지를 전송한다.
    socket.emit("chat", { msg: $msgForm.val() });
    $msgForm.val("");
  });

  function makeRandomName(){
    var name = "";
    var possible = "abcdefghijklmnopqrstuvwxyz";
    for( var i = 0; i < 3; i++ ) {
      name += possible.charAt(Math.floor(Math.random() * possible.length));
    }
    return name;
  }
});
</script>
</body>
</html>
```

io() 를 이용해 socket.io 서버에 접속

socket.emit을 통해 login정보 전송

socket.on을 통해 서버로부터의 메시지 수신

서버 실행

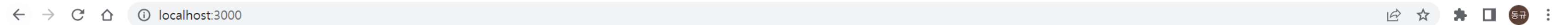
\$ node [서버 js 파일]

Ex)

```
C:\Users\<redacted>\sockettest>node server.js
Socket IO server listening on port 3000
```

클라이언트 접속

주소창에 localhost:3000

Ex) 

Socket.io Chat Example

Message:

jfs has joined

```
C:\Users\<redacted>\sockettest>node server.js
Socket IO server listening on port 3000
Client logged-in:
  name:jfs
  _userid: ungmo2@gmail.com
```

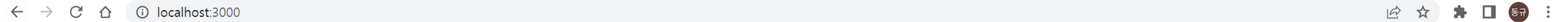
서버 실행

\$ node [서버 js 파일]

Ex) `C:\Users\ [redacted] \sockettest>node server.js`
`Socket IO server listening on port 3000`

클라이언트 접속

주소창에 localhost:3000

Ex) 

Socket.io Chat Example

Message:

jfs has joined

`C:\Users\ [redacted] \sockettest>node server.js`
`Socket IO server listening on port 3000`
`Client logged-in:`
`name:jfs`
`userid: ungmo2@gmail.com`

이벤트 송수신 테스트

Ex)

Socket.io Chat Example

Message:

hello socket.IO

Send

jfs has joined



```
C:\Users\ungmo2>cd C:\Users\ungmo2\Documents\sockettest>node server.js
Socket IO server listening on port 3000
Client logged-in:
  name:jfs
  userid: ungmo2@gmail.com
Message from jfs: test11
Message from jfs: hello socket.IO
```

The End