webOS

WebSocket

webOS OSE의 WebSocket

WebSocket 통신을 사용하는 방법

webOS

WebSocket

목표

- WebSocket의 기본 개념을 이해
- Web App에서 WebSocket을 활용하여 서버와의 실시간 데이터 교환을 구현하는 기술을 학습
- WebSocket을 이용한 실시간 통신을 구현하는 방법을 숙지

WebSocket

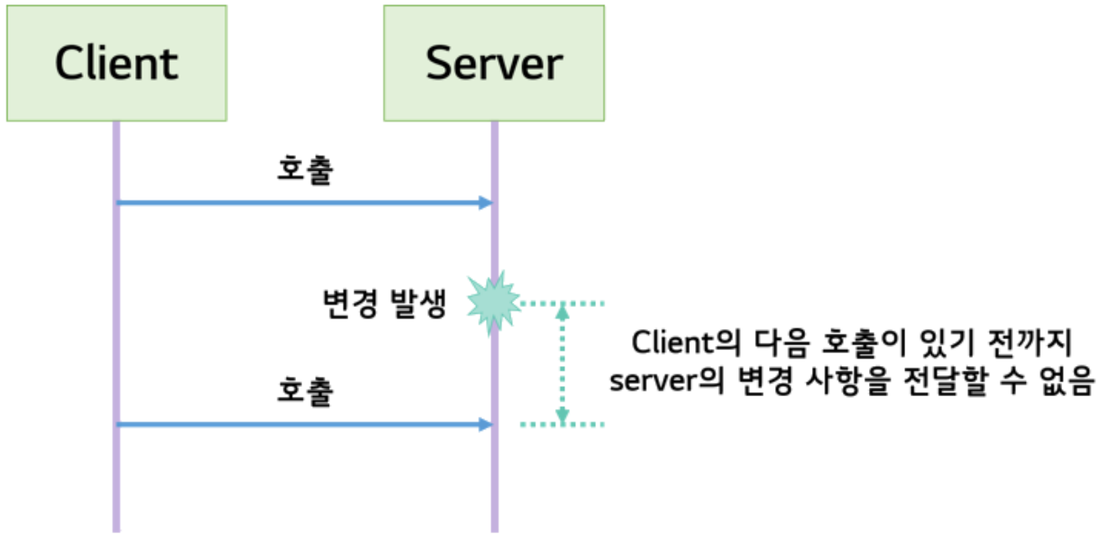
- 서버와 클라이언트 간의 통신 시 사용되는 통신 규약
- HyperText Transfer Protocol (HTTP)나 Asynchronous JavaScript And XML (AJAX)와 달리 서버 측에서도 호출할 수 있는 전이중 통신 채널을 지원

webOS

WebSocket

HTTP or AJAX

- 클라이언트 호출이 있으면 그에 대해 서버가 응답하는 방식
- 서버에서 변경 사항이 생겨도 클라이언트가 호출하기 전까지는 알 수 없으므로 실시간 통신이 어렵다는 단점이 있음

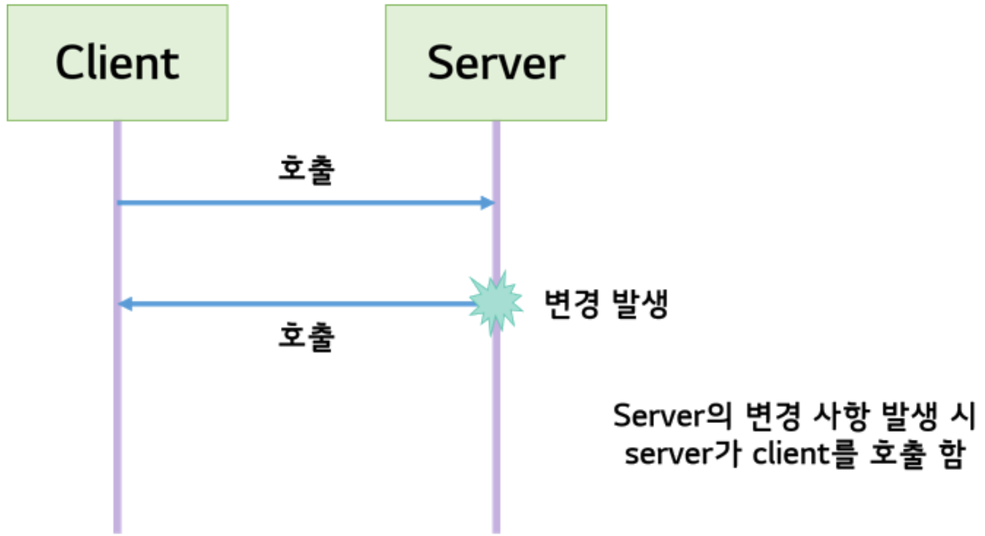


webOS

WebSocket

WebSocket

- 클라이언트와 서버 모두 서로를 호출할 수 있음
- 서버에서 전달하고자 하는 변경 사항을 발견하였을 시 즉각적으로 클라이언트에 해당 내용을 전달할 수 있음



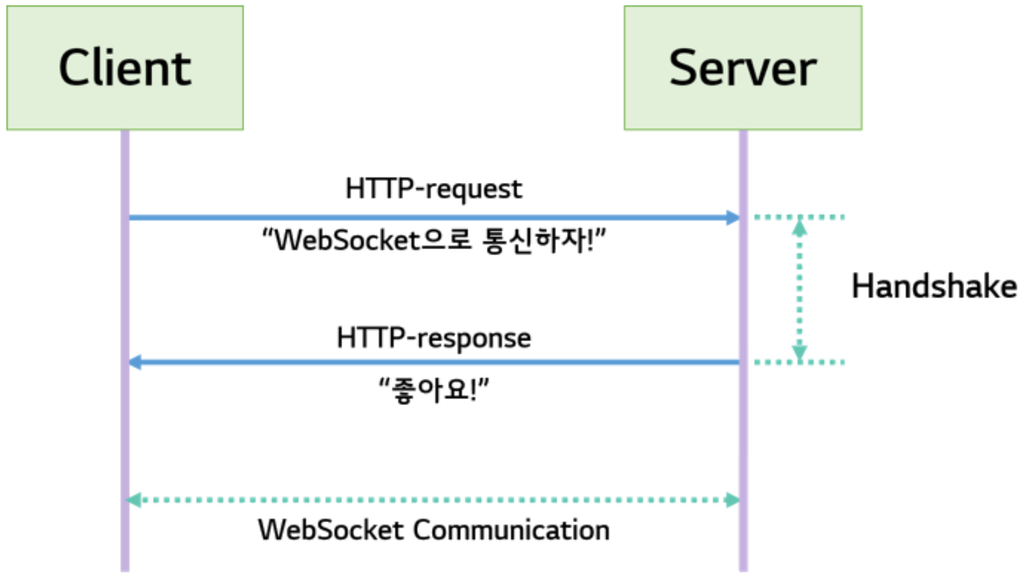
webOS

WebSocket

WebSocket 연결 방법

- WebSocket 통신하기 위해선 Handshake라는 과정을 통해 클라이언트와 서버 간의 연결을 구축
- HTTP를 이용하여 클라이언트가 서버에게 WebSocket 연결을 요청하면 서버에서 응답

```
let socket = new WebSocket("wss://webos/university-sample/websocket");
```



webOS

WebSocket 샘플 앱

샘플 – WebSocket 웹 앱 (30분)

- 시나리오
  - Node.js를 이용하여 webOS 웹 앱과 서버 간 통신 구현
- 목표
  - Node.js를 이용해 서버와 webOS 웹 앱을 동시에 사용하는 방법을 학습합니다.

webOS

WebSocket 샘플 앱

Node.js 설치 및 Node_module 설치

1. Node.js 공식 사이트를 참고하여 Node.js를 설치
2. 샘플 앱에서 사용할 모듈들을 network/websocket 폴더에서 설치

```
npm install
```

```
network/websocket
|- client.js
|- index.html
|- node_modules/
|- package-lock.json
|- package.json
|- response-window.css
|- server.js
```

3. webOS 웹 앱 설치 과정과 동일하게 패키징하여 Emulator에 설치
4. 샘플 websocket 폴더를 로컬 PC에서 Emulator 의 /home에 복사
5. shell을 통해 복사한 폴더에 접근한 뒤 아래 명령어로 서버를 실행

```
node server
```

6. 설치한 웹 앱을 실행합니다. (앱 이름: WebSocket Sample) 앱 실행과 동시에 클라이언트가 서버로의 접속을 시도

webOS

WebSocket 샘플 앱

appinfo.json

```sh
1.  {
2.    "id":"com.sample.websocket",
3.    "version":"1.0.0",
4.    "vendor":"SW Developer Experience Part",
5.    "type":"web",
6.    "main":"index.html"
7.    "title":"WebSocket Sample",
8.    "icon":"icon.png",
9.    "requiredPermission": [
10.   ]
11. }
```

- Line (2) : 앱의 ID입니다. 추후 앱을 호출할 일이 있을 때 고유한 식별자로써 사용됩니다.
- Line (5) : 앱의 종류입니다. 본 샘플 앱은 웹 앱이므로 web으로 설정하였습니다.
- Line (6) : 앱의 실행 시 시작 지점입니다. 웹 앱 실행 시 시작 지점으로 지정된 파일의 내용을 렌더링하여 화면에 보여줍니다.

webOS

WebSocket 샘플 앱

server.js

- WebSocket 서버의 동작을 설정

```sh
const WebSocket = require('ws');
const cron = require('node-cron');

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

let clientCnt = 0;

wss.on('connection', (wsClient, handshakeRequest) => {

// 접속한 client 수를 통해 클라이언트별 고유 ID를 생성
const clientId = clientCnt;
clientCnt += 1;

const clientIp = handshakeRequest.headers['x-forwarded-for'] ||
handshakeRequest.socket.remoteAddress;

// 최초 연결 시 클라이언트로 메시지를 보냄
wsClient.send("Hello, WebSocket client! It's your server! Your client ID is " + clientId);
console.log("====================================");
console.log("WebSocket Connection is established!");
console.log("IP: " + clientIp);
console.log("====================================");

```

webOS

WebSocket 샘플 앱

server.js

- WebSocket 서버의 동작을 설정

```sh
// 10초 마다 연결 유지 시간 출력
let elapsedTime = 10;
let tenSecsReminder = cron.schedule('*/10 * * * * *', function () {

  if (elapsedTime == 300) { // 300초 이상 연결 유지 시 종료
    tenSecsReminder.stop();
    wsClient.close();
  }

  console.log("Client " + clientId + " 접속 시간 " + elapsedTime + "초 경과");
  wsClient.send("Client " + clientId + " 접속 시간 " + elapsedTime + "초 경과");

  elapsedTime += 10;
},
{
  schedule: false
});

// 연결 유지 시간 출력 프로세스 시작
tenSecsReminder.start();

wsClient.on('close', () => {
  console.log('Client: [' + clientId + '] is disconnected.');
  tenSecsReminder.stop();
});
```

webOS

WebSocket 샘플 앱

server.js

- WebSocket 서버의 동작을 설정

```sh
48.
49.   wsClient.on('message', message => {
50.     const msgFromClient = message;
51.
52.     console.log("Message from Client[" + clientId + "]:" + msgFromClient);
53.   });
54. });
```

webOS

WebSocket 샘플 앱

client.js

- WebSocket 클라이언트의 동작을 구성하는 파일

```sh
window.onload = function () {
  const url = 'ws://localhost:8080';
  const ws = new WebSocket(url);

  ws.onopen = () => {
    ws.send("Hi Server! It's client!");
  };

  ws.onerror = (error) => {
    console.log("WebSocket error: " + error);
  };

  ws.onmessage = (message) => {
    console.log(message.data);
    printMessage(message.data);
  };

  ws.onclose = function(event) {
    if (event.wasClean) {
      printMessage("WebSocket connection is closed normally. (code: " + event.code + ", reason: " + event.reason + "}");
    } else {
      printMessage("Something wrong!");
    }
  }

```

webOS

WebSocket 샘플 앱

client.js

- WebSocket 클라이언트의 동작을 구성하는 파일

```sh
26.  function printMessage(msg) {
27.    let responseWindow = document.getElementById('client-window');
28.    responseWindow.innerHTML += msg + "<br />";
29.  }
30.
31.  // putKind method 호출을 위한 event listener
32.  const sendMessage = document.getElementById("message-btn");
33.  sendMessage.addEventListener('click', function (e) { // 'e'란 무엇인가?
     (https://stackoverflow.com/questions/35936365/what-exactly-is-the-parameter-e-event-and-why-pass-it-to-javascript-functions)
34.    console.log("Send button is clicked. Send a message to the server!");
35.
36.    const message = document.getElementById("message-input").value;
37.    ws.send(message);
38.  });
39. }
```

Question