

node.js 에서 socket.io 사용하기

장태성

http://forum.falinux.com/zbxe/index.php?document_srl=573508

2012.09.15 14:29:24 (*.52.177.29)

39812

node.js 에서 socket.io 사용하기

npm을 통해서 node.js에서 사용할 수 있는 패키지 모듈을 설치해보고 사용해보는 예제를 테스트 해볼까 합니다.

웹 소켓 통신에 주로 쓰이는 socket.io 의 샘플을 다뤄볼까합니다.



1. socket.io 란?

지금까지 웹 브라우저 통신을 한다고 하면 불편한 점이 많았습니다.

http 프로토콜을 가지고 데이터를 주고 받거나 php에서 url을 넘겨서 데이터를 교환하고자 했는데 구현할 때 불편함을 많이 느꼈습니다.

이번에 html5 가 등장하면서 실시간 웹 소켓 통신인 Websocket을 내놓았지만, 표준화 되기 까지 아직 부족한 부분이 많습니다.

현재 다양한 브라우저에서 지원을 하고 기술적으로 발달하고 있지만, 시범적으로 써볼 단계일 뿐 널리 사용되고 편하게 사용되고 있지 않습니다.

websocket이 아직 개발중인 단계라면 socket.io 는 node.js에서 바로 사용할 수 있는 기술입니다.

엄밀히 말하면 socket.io가 아직 표준 기술은 아니지만, javascript를 사용하면 node.js에서

외부의 다른 통신요소를 거치지 않고 실시간 웹을 구현할 수 있는 기술이라서 유용하게 쓰일 수 있습니다.

node.js 에서 소켓 통신을 지원하는 모듈이 몇가지 더 있지만, 가장 안정적이고 편하게 사용할 수 있는게 socket.io 입니다.

2. socket.io 설치

지난 주에 설명했던 npm을 가지고 원하는 모듈을 설치할 수 있습니다.

socket.io 홈페이지에 접속하면 기본적인 설명과 샘플을 볼 수 있습니다.

홈페이지 : <http://socket.io>

아래 그림은 첫 페이지 화면입니다.



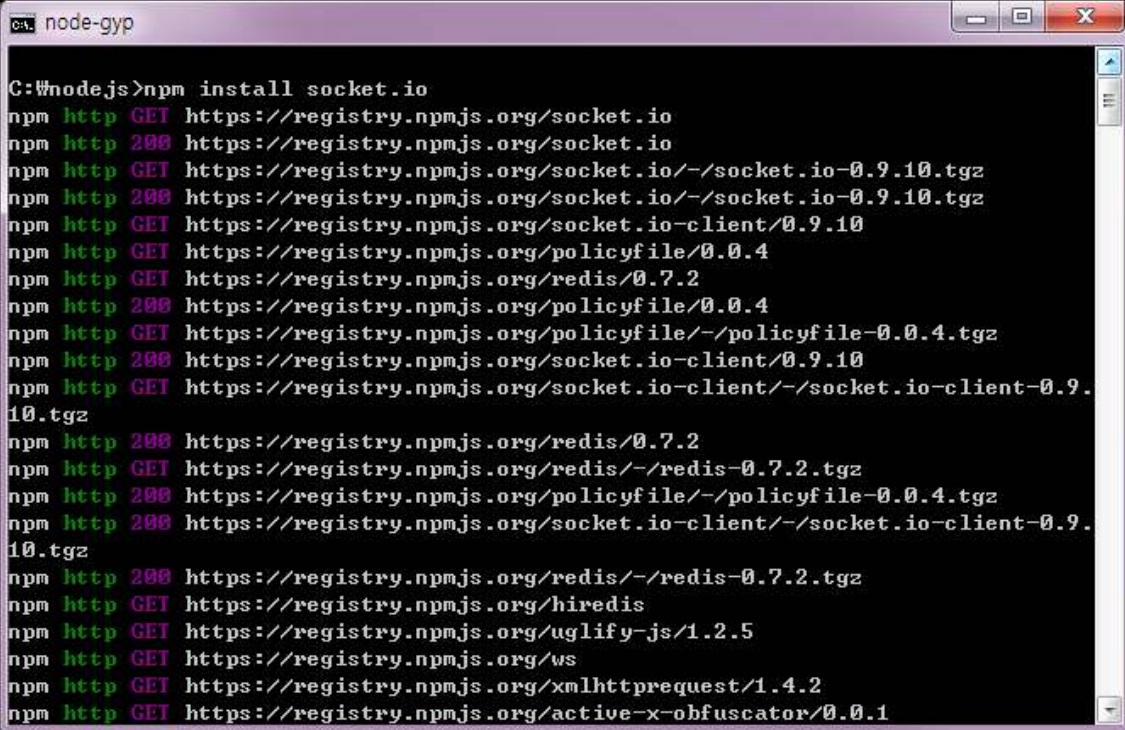
우선 설치를 하려면 npm을 사용하면 됩니다.

편의상 윈도우 환경에서 작업을 했으며, C 드라이브에 nodejs 라는 폴더를 만든 후에 작업을 진행하였습니다.

아래와 같이 npm 명령을 입력합니다.

```
c:\nodejs>npm install socket.io
```

아래 그림과 같이 npm 패키지를 관리하는 곳에서 socket.io의 정보를 받아와 다운로드 하게 됩니다.



```
C:\nodejs>npm install socket.io
npm http GET https://registry.npmjs.org/socket.io
npm http 200 https://registry.npmjs.org/socket.io
npm http GET https://registry.npmjs.org/socket.io/-/socket.io-0.9.10.tgz
npm http 200 https://registry.npmjs.org/socket.io/-/socket.io-0.9.10.tgz
npm http GET https://registry.npmjs.org/socket.io-client/0.9.10
npm http GET https://registry.npmjs.org/policyfile/0.0.4
npm http GET https://registry.npmjs.org/redis/0.7.2
npm http 200 https://registry.npmjs.org/policyfile/0.0.4
npm http GET https://registry.npmjs.org/policyfile/-/policyfile-0.0.4.tgz
npm http 200 https://registry.npmjs.org/socket.io-client/0.9.10
npm http GET https://registry.npmjs.org/socket.io-client/-/socket.io-client-0.9.10.tgz
npm http 200 https://registry.npmjs.org/redis/0.7.2
npm http GET https://registry.npmjs.org/redis/-/redis-0.7.2.tgz
npm http 200 https://registry.npmjs.org/policyfile/-/policyfile-0.0.4.tgz
npm http 200 https://registry.npmjs.org/socket.io-client/-/socket.io-client-0.9.10.tgz
npm http 200 https://registry.npmjs.org/redis/-/redis-0.7.2.tgz
npm http GET https://registry.npmjs.org/hiredis
npm http GET https://registry.npmjs.org/uglify-js/1.2.5
npm http GET https://registry.npmjs.org/ws
npm http GET https://registry.npmjs.org/xmlhttprequest/1.4.2
npm http GET https://registry.npmjs.org/active-x-obfuscator/0.0.1
```

```
C:\ 관리자: C:\Windows\system32\cmd.exe
npm http 200 https://registry.npmjs.org/commander
npm http GET https://registry.npmjs.org/commander/-/commander-0.6.1.tgz
npm http 200 https://registry.npmjs.org/zeparser/-/zeparser-0.0.5.tgz
npm http 200 https://registry.npmjs.org/options/-/options-0.0.3.tgz
npm http 200 https://registry.npmjs.org/tinycolor/-/tinycolor-0.0.1.tgz
npm http 200 https://registry.npmjs.org/commander/-/commander-0.6.1.tgz

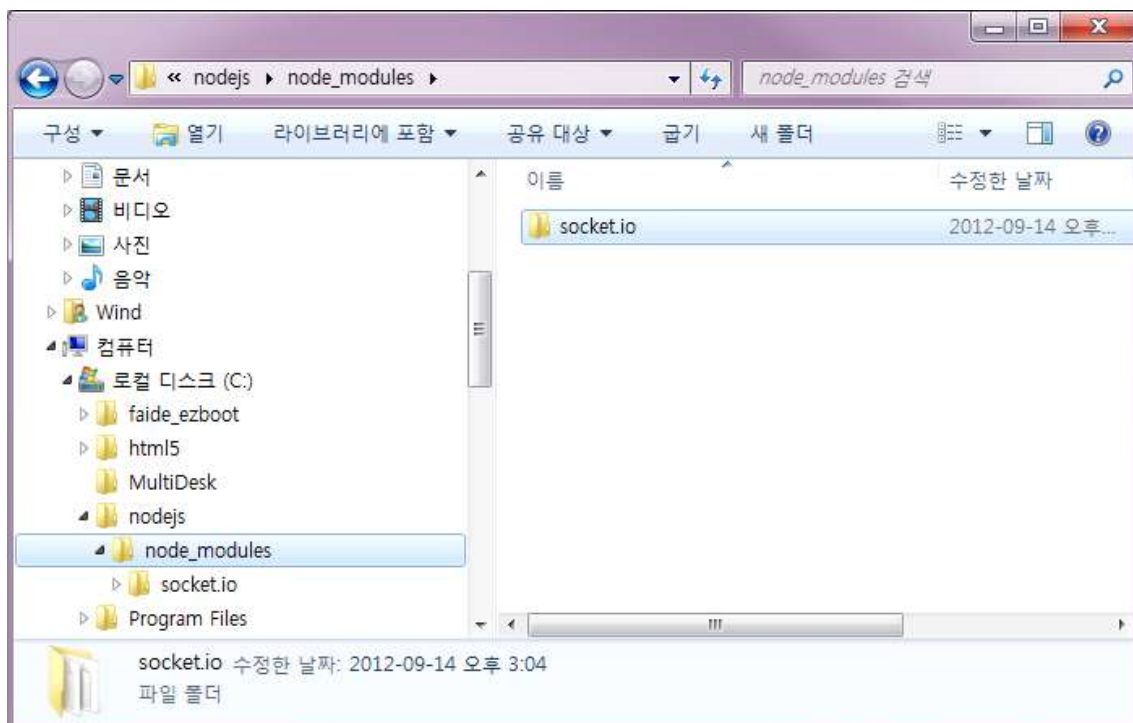
> ws@0.4.21 install C:\nodejs\node_modules\socket.io\node_modules\socket.io-client\node_modules\ws
> node install.js

lib v0.4.21 Attempting to compile blazing fast native extensions.
lib v0.4.21 Native code compile failed (but the module will still work):
lib v0.4.21 The native extensions are faster, but not required.
lib v0.4.21 On Windows, native extensions require Visual Studio and Python.
lib v0.4.21 On Unix, native extensions require Python, make and a C++ compiler.

lib v0.4.21 Start npm with --ws:verbose to show compilation output (if any).
socket.io@0.9.10 node_modules\socket.io
├── policyfile@0.0.4
├── redis@0.7.2
├── socket.io-client@0.9.10 (xmlhttprequest@1.4.2, uglify-js@1.2.5, ws@0.4.21)
└── active-x-obfuscator@0.0.1

C:\nodejs>
```

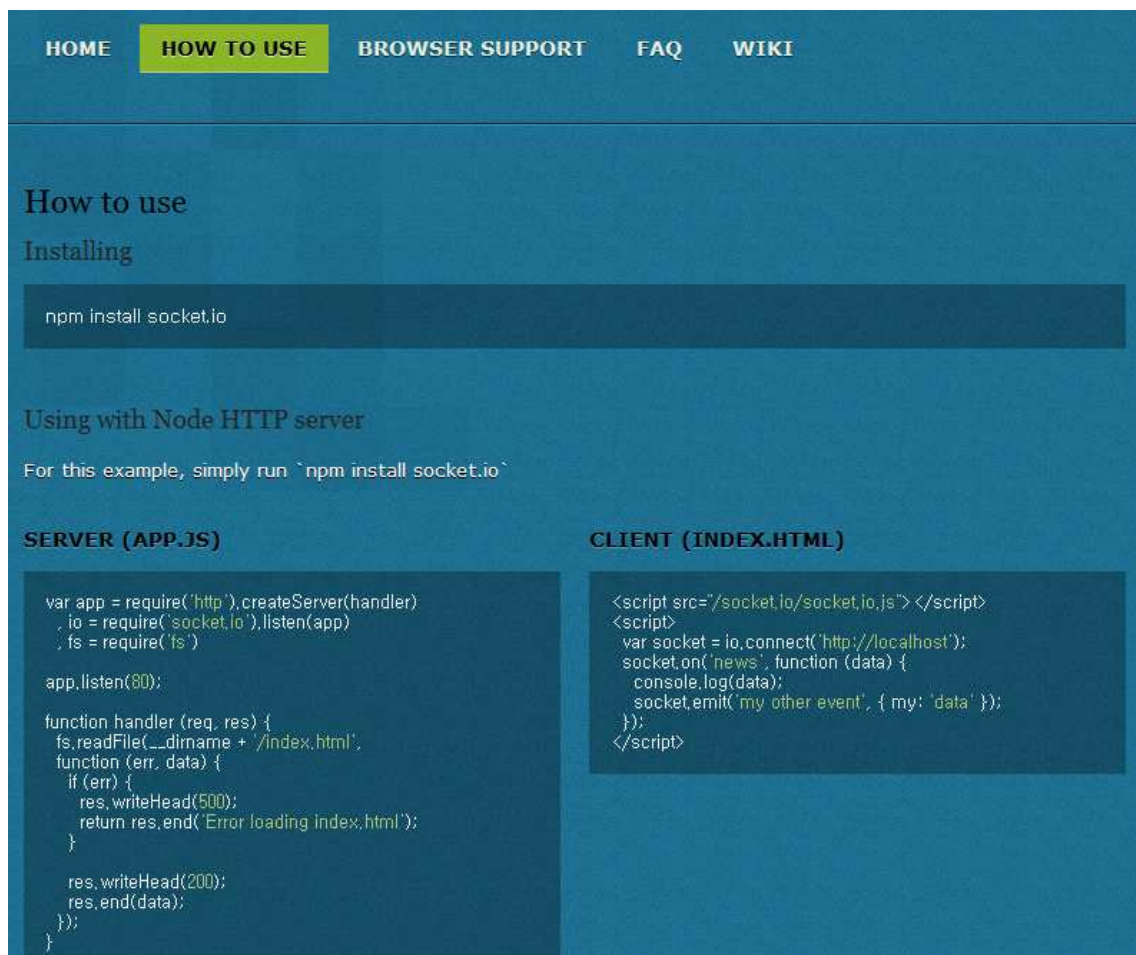
설치가 완료되면 아래와 같이 node_modules 이라는 폴더가 생기고 그 안에 설치한 패키지가 생깁니다.



3. 서버 프로그램 작성

패키지를 설치하였으니, 서버 프로그램을 작성해볼 차례입니다.

socket.io 홈페이지에 가면 HOW To USE 에 가면 아래와 같이 샘플 예제를 보여줍니다.



The screenshot shows the 'How to use' section of the socket.io website. It includes a navigation bar with links to HOME, HOW TO USE (highlighted), BROWSER SUPPORT, FAQ, and WIKI. The main content area is titled 'How to use' and 'Installing'. It provides the command 'npm install socket.io'. Below this, it shows 'Using with Node HTTP server' with the instruction 'For this example, simply run `npm install socket.io`'. The page is divided into two columns: 'SERVER (APP.JS)' and 'CLIENT (INDEX.HTML)'. The server column contains a JavaScript code snippet for a simple HTTP server using socket.io. The client column contains a JavaScript code snippet for a client connecting to the server and emitting an event.

```
var app = require('http').createServer(handler)
    , io = require('socket.io').listen(app)
    , fs = require('fs')

app.listen(80);

function handler (req, res) {
  fs.readFile(__dirname + '/index.html',
    function (err, data) {
      if (err) {
        res.writeHead(500);
        return res.end('Error loading index.html');
      }

      res.writeHead(200);
      res.end(data);
    });
}
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
  });
</script>
```

그대로 해보도록 합니다.

서버 프로그램은 node.js를 통해서 실행할 것이므로 파일의 확장자는 js 로 만들어 줍니다.

C 드라이브 nodejs 폴더 아래 app.js 라는 파일을 만들어서 아래 내용을 입력해줍니다.

```

var app = require('http').createServer(handler)
  , io = require('socket.io').listen(app)
  , fs = require('fs')

app.listen(80);

function handler (req, res) {
  fs.readFile(__dirname + '/index.html',
    function (err, data) {
      if (err) {
        res.writeHead(500);
        return res.end('Error loading index.html');
      }

      res.writeHead(200);
      res.end(data);
    });
}

io.sockets.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data) {
    console.log(data);
  });
});

```

node.js에서 모듈을 사용할 때는 require를 통해서 사용하고자 하는 모듈명을 적어줍니다. 실제 모듈 위치는 node_modules 아래 socket.io 이지만, node_modules는 node.js에서 자동으로 관리하기 때문에 적어주지 않고 모듈명만 적어주도록 합니다.

기본적인 웹서버를 기동하기 위해서 http를 사용해서 서버를 생성해줍니다.

```
var app = require('http').createServer
```

아래 부분은 socket.io 의 가장 기본적인 함수입니다.

socket.io를 활성화 하고 외부의 클라이언트에서 접속, 즉 connection을 하면 명령을 수행한

다는 뜻입니다.

접속을 하게 되면 무조건 저 함수를 거치게 된다고 보면 됩니다.

```
io.sockets.on('connection', function (socket) {  
  }  
}
```

socket.emit 는 접속한 대상에게 데이터를 전송하는 부분입니다.

위의 내용에 따르면 서버에 접속한 클라이언트에게 바로 'news' 라는 내용을 전송합니다.

```
socket.emit
```

아래는 데이터 부분인데, node.js 에서는 기본적으로 json 방식으로 데이터를 보냅니다.

```
'news', { hello: 'world' }
```

on 은 명령 또는 데이터를 받기 위해 대기하고 있는 것입니다.

예제에 따라 서버에서는 대기하고 있다가 클라이언트에서 'my other event' 라는 데이터를 받게 되면 함수 내용을 수행하게 됩니다.

```
socket.on
```

4. 클라이언트 프로그램 작성

서버 프로그램을 작성했으니, 클라이언트 프로그램을 작성할 차례입니다.

클라이언트 화면에 보여질 부분, 즉 웹 브라우저에 나타날 부분이라서 html 파일을 만들어줍니다.

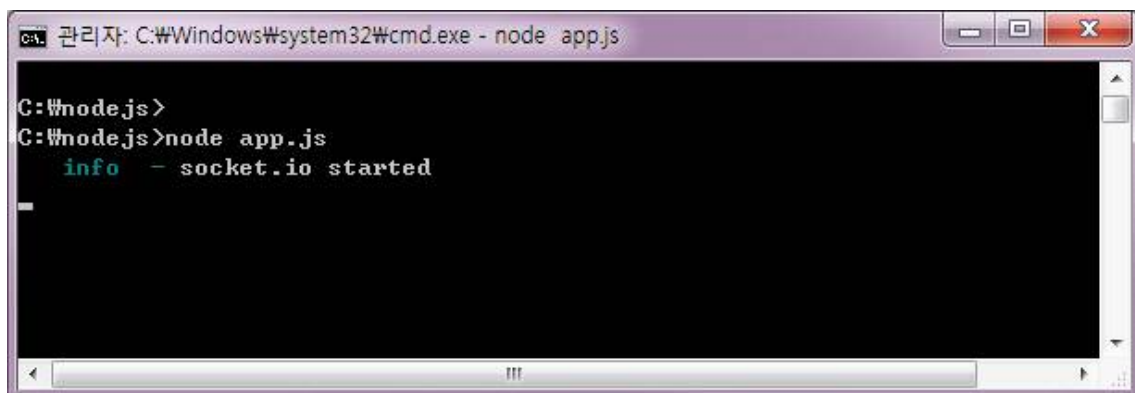
index.html 이라는 파일을 만들어서 아래 내용을 입력하였습니다.


```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
  });
</script>
```

5. 프로그램 테스트

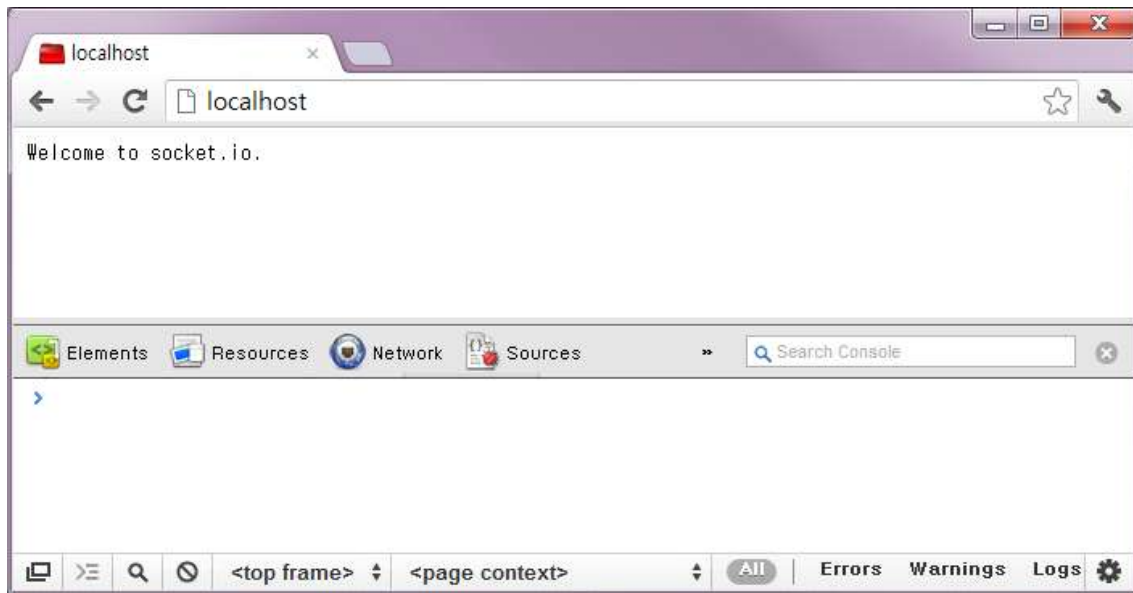
프로그램 테스트는 먼저 서버 프로그램을 실행해줍니다.

아래와 같이 node app.js 라고 명령을 입력해주면 node.js 가 실행됩니다.

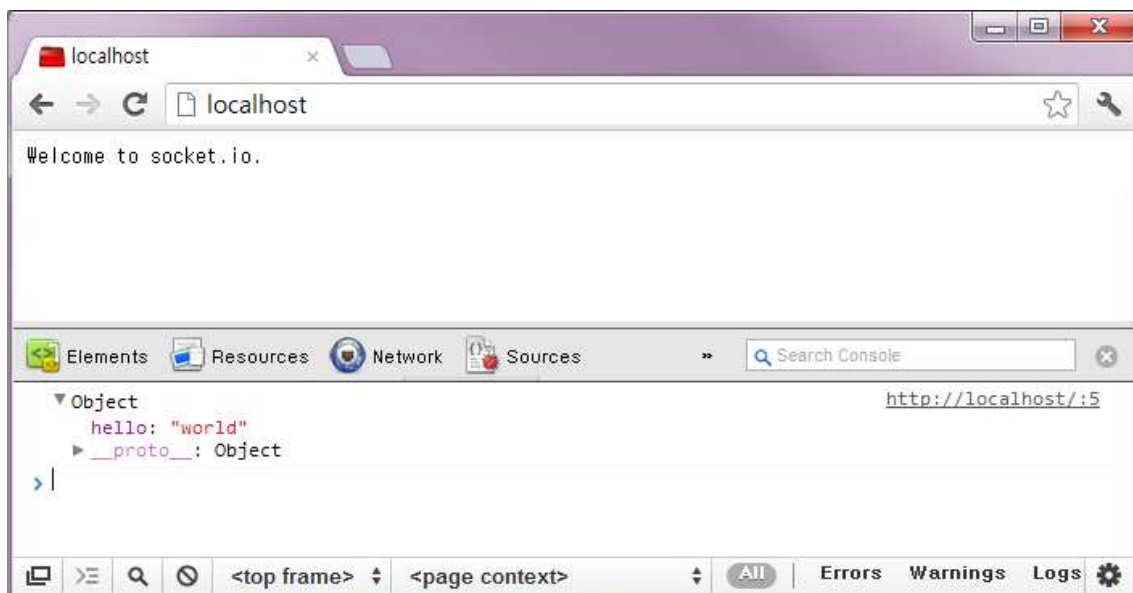


서버는 현재 대기중이기 때문에 동작을 보려면 웹 브라우저에 접속을 합니다.

html5를 가장 많이 지원하는 크롬 브라우저를 선호하는 편이라 웹 브라우저를 띄우고
http://localhost 로 접속을 합니다.

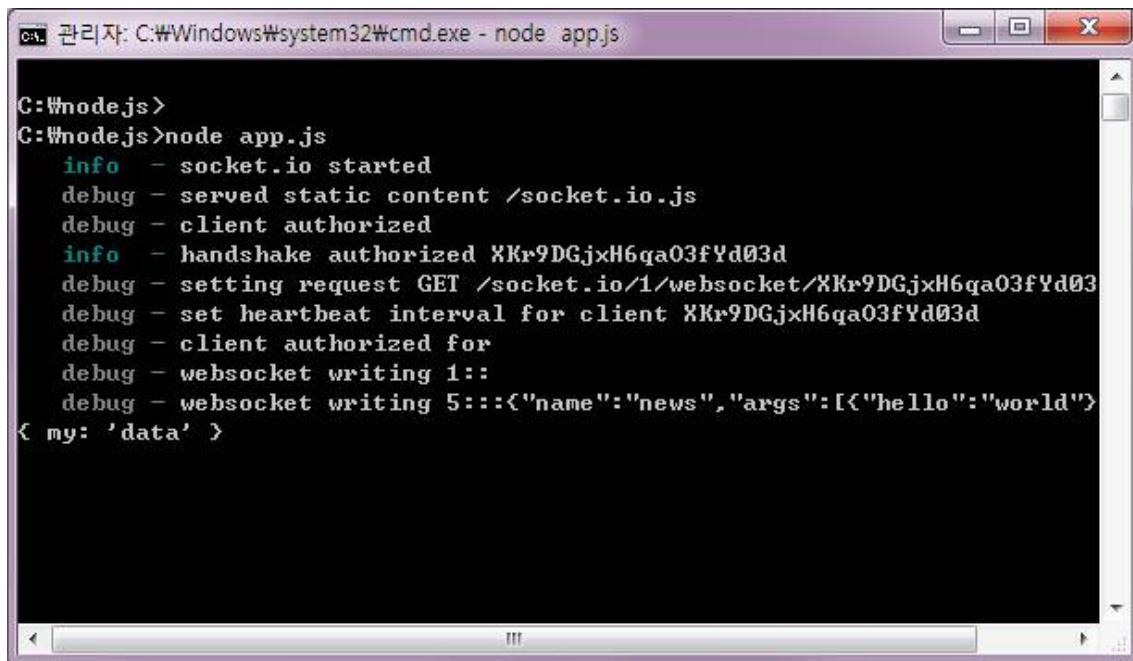


접속을 하면 대기하고 있던 서버에서는 'news' 데이터를 웹 브라우저에 보냅니다.
그러면 웹 브라우저에서는 'news' 라는 데이터를 받고 그에 해당하는 함수를 실행합니다.
먼저 받은 데이터를 console.log 메시지를 사용하여 출력합니다.



그 다음 socket.emit을 통해서 서버로 'my other event', { my: 'data' } 데이터를 보냅니다.
서버에서는 'my other event'가 오기 만을 기다렸다가 데이터를 받아서 { my: 'data' } 데이

터를 출력합니다.



```
C:\Windows\system32\cmd.exe - node app.js
C:\nodejs>
C:\nodejs>node app.js
  info - socket.io started
  debug - served static content /socket.io.js
  debug - client authorized
  info - handshake authorized XKr9DGjxH6qa03fYd03d
  debug - setting request GET /socket.io/1/websocket/XKr9DGjxH6qa03fYd03d
  debug - set heartbeat interval for client XKr9DGjxH6qa03fYd03d
  debug - client authorized for
  debug - websocket writing 1::
  debug - websocket writing 5::{"name":"news","args":[{"hello":"world"}]}
{ my: 'data' }
```

~(-_ - ~)

샘플에서는 서버가 먼저 실행되어 대기하고 있고, 클라이언트에서 접속을 할 때, 메시지를 서로 주고 받는 구조로 되어 있습니다. 위의 방법을 활용하여, 버튼의 액션으로 내용을 전송하거나 타이머와 같이 주기적으로 요청을 하여 값을 받아올 수 있습니다.

node.js 안에서 되는 소켓 통신 socket.io 에 대해서 살펴보았습니다.~