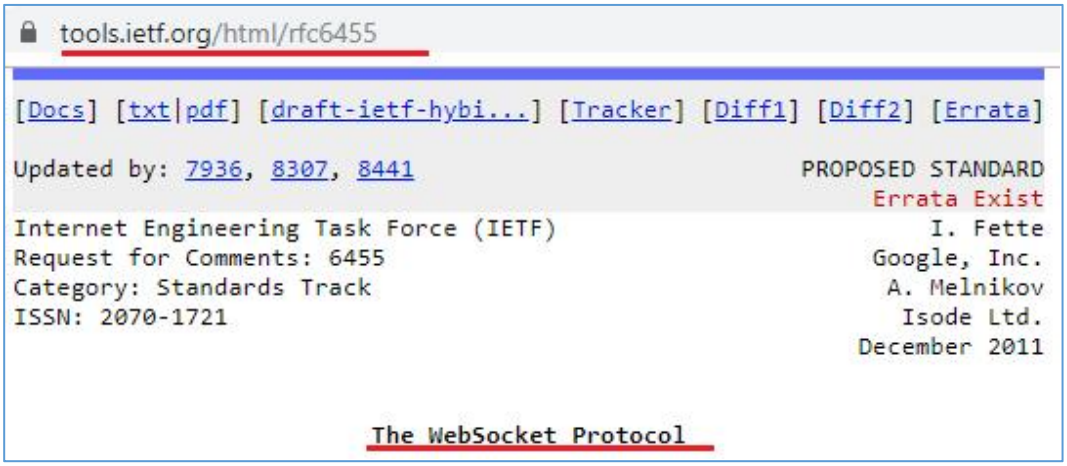
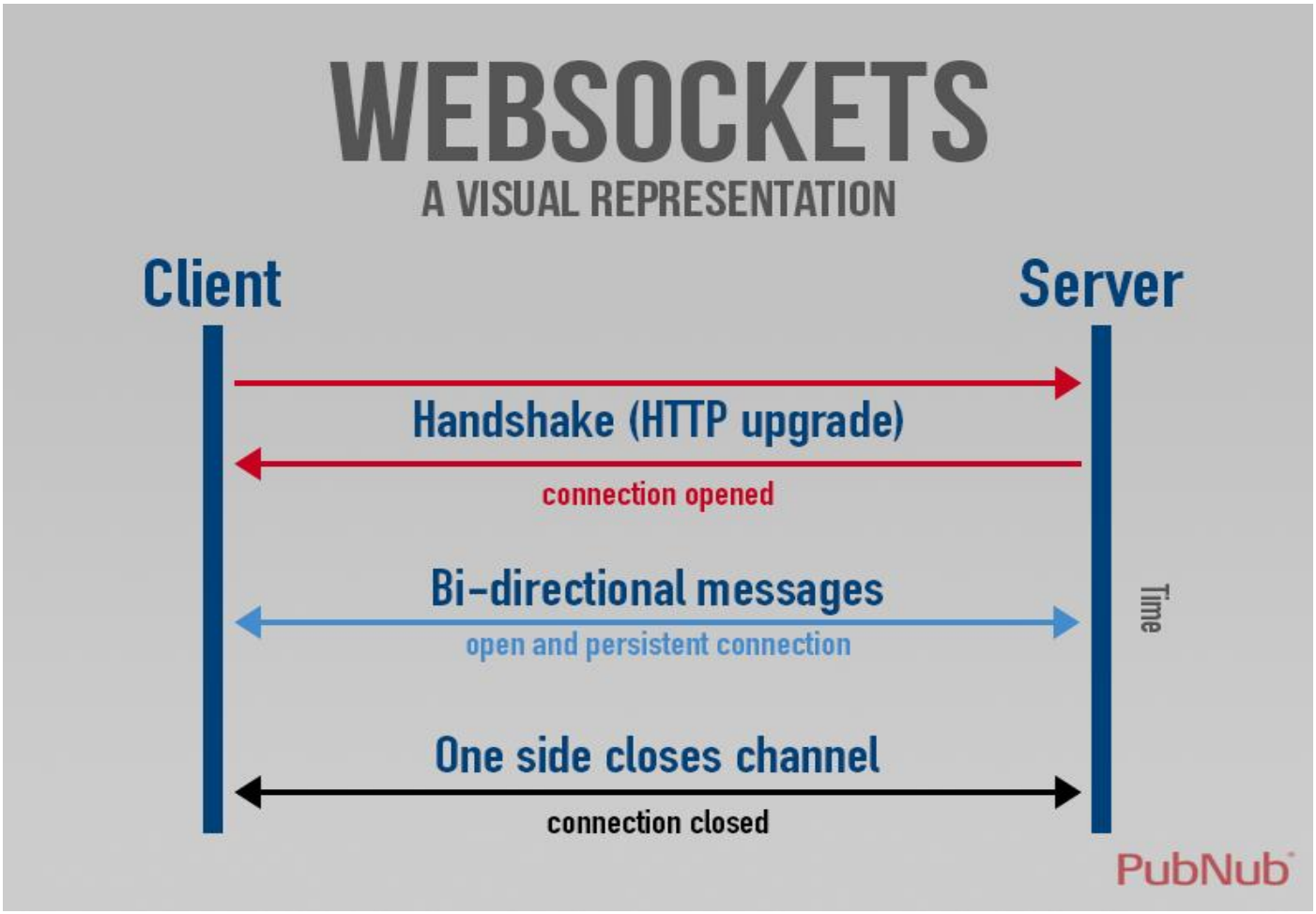
Node.js-WebSockets-сервер

1. **WebSockets:** https://tools.ietf.org/html/rfc6455
2. **WebSockets:** https://learn.javascript.ru/websockets





3. **WebSockets:** **handshake**
4. **WebSockets:** https://github.com/websockets/ws
5. **WebSockets:** `npm install ws`
6. **WebSockets:** `сервер, HTML5-браузер`

```
const httpserver = require('http').createServer((req, res)=>{
    if (req.method == 'GET' && req.url == '/start'  ){
        res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
        res.end(require('fs').readFileSync('./09-01.html'));
    }
});
httpserver.listen(3000)
console.log ('ws server: 3000');
```

```
let k = 0;
const WebSocket = require('ws')   // npm install ws
const wsserver = new WebSocket.Server({ port: 4000, host:'localhost', path:'/wsserver'})
wsserver.on('connection', (ws) => {
        ws.on('message', message => {
            console.log(`Received message => ${message}`)
        })
        setInterval(()=>{ws.send(`server: ${++k}`)}, 1500);
})
wsserver.on('error',(e)=>{console.log('ws server error', e)});
console.log(`ws server: host:${wsserver.options.host}, port:${wsserver.options.port}, path:${wsserver.options.path}`);
```

```
<body>
<h1>Lec 09</h1>
      <script>
            let k = 0;
            function startWS(){

                  let socket = new WebSocket('ws:/localhost:4000/wsserver');

                  socket.onopen = ()=>{ console.log('socket.onopen');
                        setInterval(()=>{socket.send(++k);}, 1000);
                  };

                  socket.onclose =  (e)=>{ console.log('socket.onclose', e);};

                  socket.onmessage =(e)=>{console.log('socket.onmessage',  e.data)};

                  socket.onerror = function(error) { alert("Ошибка " + error.message); };

            };
      </script>
      <button  onclick="startWS()">startWS</button>
</body>
```

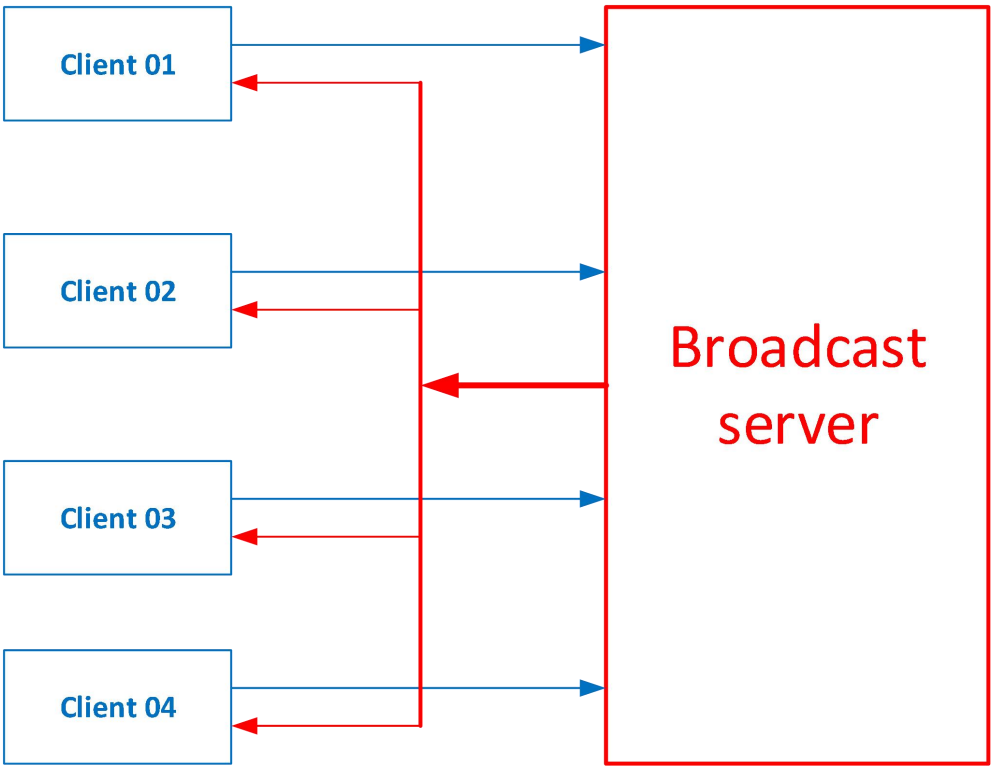
7. **WebSockets:** клиент

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

const ws = new WebSocket('ws://localhost:4000/wsserver');

ws.on('open', () =>{
    ws.send('message 1');    // отправить сообщение серверу
    ws.send('message 2');    // отправить сообщение серверу
    ws.on('message', message => {
        console.log(`Received message => ${message}`)
    })
    setTimeout(()=>{ws.close()},5000);  // остановить через 5 секуд
});
```

8. **WebSockets:** сервер, broadcast, уведомления



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

const wss = new WebSocket.Server({ port: 5000, host:'localhost', path:'/broadcast' });

wss.on('connection', (ws)=>{
        ws.on('message', (data)=>{

            wss.clients.forEach((client)=>{
                    if (client.readyState === WebSocket.OPEN) client.send('server: '+data);
            });

        });
});
```

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

  let parm0 = process.argv[0];    // path node
  let parm1 = process.argv[1];    // path application
  let parm2 = process.argv[2];    // first parameter

  console.log('parm2 = ', parm2);

  let prfx = typeof parm2 == 'undefined'?'A':parm2;
  const ws = new WebSocket('ws://localhost:5000/broadcast');

  ws.on('open', () =>{

      let k = 0;
      setInterval(()=>{
          ws.send(`client: ${prfx}-${++k}`);        // отправить сообщение серверу
      }, 1000);

      ws.on('message', message => {
          console.log(`Received message => ${message}`)
      })

      setTimeout(()=>{ws.close()},25000);  // остановить через 25 секуд
  });
```

9. **WebSockets:** клиент, взаимодействие с потоком

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

const ws = new WebSocket('ws://localhost:5000/broadcast');

const duplex = WebSocket.createWebSocketStream(ws, { encoding: 'utf8' });

duplex.pipe(process.stdout);  //  сообщения от сервера --> stdout

process.stdin.pipe(duplex);   //  stdin --> сообщение серверу
```

10. **WebSockets:** ping/pong, server-client

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 5000, host:'localhost'});
wss.on('connection', (ws)=>{
            ws.on('pong', (data)=>{
                console.log('on pong: ', data.toString());
            });
            ws.on('message', (data)=>{
                console.log('on message: ', data.toString());
                ws.send(data);
            });
            setInterval(()=>{ console.log('server: ping');  ws.ping('server: ping')}, 5000);
});
```

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

const ws = new WebSocket('ws://localhost:5000');

const duplex = WebSocket.createWebSocketStream(ws, { encoding: 'utf8' });

duplex.pipe(process.stdout);  //  сообщения от сервера --> stdout

process.stdin.pipe(duplex);   //  stdin --> сообщение серверу

// ws.on('ping', (data)=>{    // можно ловить
//     console.log('on ping: ', data.toString());
// });
```

11. **WebSockets:** ping/pong, client-server

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 5000, host:'localhost'});
wss.on('connection', (ws)=>{

        ws.on('message', (data)=>{
            console.log('on message: ', data.toString());
            ws.send(data);
        });

});
wss.on('error',(e)=>{console.log('wss server error', e)});
```

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

const ws = new WebSocket('ws://localhost:5000');

const duplex = WebSocket.createWebSocketStream(ws, { encoding: 'utf8' });

duplex.pipe(process.stdout);  //  сообщения от сервера --> stdout

process.stdin.pipe(duplex);   //  stdin --> сообщение серверу

ws.on('pong', (data)=>{   // можно ловить
    console.log('on pong: ', data.toString());
});
setInterval(()=>{ console.log('server: ping');  ws.ping('client: ping')}, 5000);
```

12. **WebSockets:** JSON

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 5000, host:'localhost'});
wss.on('connection', (ws)=>{

        ws.on('message', (data)=>{console.log('on message: ', JSON.parse(data)); });

        let k = 0;
        setInterval(()=>{ ws.send(JSON.stringify({k:++k, sender:'Server', date: new Date().toISOString()}));}, 5000);

});
```

```
const WebSocket = require('ws');
const ws = new WebSocket('ws://localhost:5000');
ws.on('open', () =>{
    ws.on('message', data => { console.log('on message: ', JSON.parse(data)); })
    let k = 0;
    setInterval(()=>{ws.send(JSON.stringify({k:++k, sender:'Client',  date: new Date().toISOString()}));}, 3000);
});
```

13. **WebSockets:** upload file

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

const wss = new WebSocket.Server({ port: 5000, host:'localhost'});
let k = 0;
wss.on('connection', (ws)=>{
        const duplex = WebSocket.createWebSocketStream(ws, { encoding: 'utf8' });
        let wfile =  fs.createWriteStream(`./file${++k}.txt`);
        duplex.pipe(wfile);
});
```

```
const fs =  require('fs');
const WebSocket = require('ws');
const ws = new WebSocket('ws://localhost:5000');
ws.on('open', () =>{
    const duplex = WebSocket.createWebSocketStream(ws, { encoding: 'utf8' });
    let rfile =  fs.createReadStream(`./MyFile.txt`);
    rfile.pipe(duplex);
});
```

14. **WebSockets:** download file

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

const wss = new WebSocket.Server({ port: 5000, host:'localhost'});
wss.on('connection', (ws)=>{
        const duplex = WebSocket.createWebSocketStream(ws, { encoding: 'utf8' });
        let rfile =  fs.createReadStream('./MyFile.txt');
        rfile.pipe(duplex);
});
```

```
const fs =  require('fs');
const WebSocket = require('ws');
const ws = new WebSocket('ws://localhost:5000');
let k = 0;
ws.on('open', () =>{
    const duplex = WebSocket.createWebSocketStream(ws, { encoding: 'utf8' });
    let wfile =  fs.createWriteStream(`./MyFile${++k}.txt`);
    duplex.pipe(wfile);
});
```

15. **WebSockets:** npm install **rpc-WebSocketss**
16. **WebSockets:** rpc-websockets, RPC, public, ***клиент не завершается***

```
const rpcWSS = require('rpc-websockets').Server

let server = new rpcWSS({port: 5000,  host: 'localhost'});

server.register('sum', (params) =>{ return params[0] + params[1]}).public();
server.register('sub', (params) =>{ return params[0] - params[1]}).public();
server.register('mul', (params) =>{ return params[0] * params[1]}).public();
server.register('div', (params) =>{ return params[0] / params[1]}).public();
server.register('get', (params) =>{
                    // сходить в БД за данными
                    return {id:params[0], name:'Иванов И.И.', bday: '2000-12-02'};
      }).public();
```

```
const rpcWSC = WebSocket = require('rpc-websockets').Client
let ws = new rpcWSC('ws://localhost:5000');

ws.on('open', ()=>{
    ws.call('sum', [5, 3]).then((r)=>{console.log('sum = ', r);});
    ws.call('sub', [5, 3]).then((r)=>{console.log('sub = ', r);});
    ws.call('mul', [5, 3]).then((r)=>{console.log('mul = ', r);});
    ws.call('div', [5, 3]).then((r)=>{console.log('div = ', r);});
    ws.call('get', [123], 3000).catch((e)=>{return e;}).then((r)=>{console.log('get_student = ', r);});
});
ws.on('error', (e)=>{console.log('error = ', e) });
```

17. **WebSockets:** rpc-websockets, RPC, protected, ***клиент не завершается***

```
const rpcWSS = require('rpc-websockets').Server

let server = new rpcWSS({port: 5000,  host: 'localhost'});

server.setAuth((l)=>{ return (l.login == 'smw' &&  l.password == '777') });


server.register('sum', (params) =>{ return params[0] + params[1]}).public();
server.register('sub', (params) =>{ return params[0] - params[1]}).public();
server.register('mul', (params) =>{ return params[0] * params[1]}).public();
server.register('div', (params) =>{ return params[0] / params[1]}).public();
server.register('get', (params) =>{return {id:params[0], name:'Иванов И.И.', bday: '2000-12-02'};}).public();

server.register('mod', (params) =>{ return params[0] % params[1]}).protected();
server.register('abs', (params) =>{ return params[0] >= 0?params[0]: -params[0]}).protected();
```

```
const rpcWSC = WebSocket = require('rpc-websockets').Client
let ws = new rpcWSC('ws://localhost:5000');

ws.on('open', ()=>{
    ws.call('sum', [5, 3]).then((r)=>{console.log('sum = ', r);});
    ws.call('sub', [5, 3]).then((r)=>{console.log('sub = ', r);});
    ws.call('mul', [5, 3]).then((r)=>{console.log('mul = ', r);});
    ws.call('div', [5, 3]).then((r)=>{console.log('div = ', r);});
    ws.call('get', [123], 3000).catch((e)=>{return e;}).then((r)=>{console.log('get_student = ', r);});

    ws.login({login: 'smw', password: '787'})
      .then ((login) =>{
                if (login){

                  ws.call('mod', [33, 5]).then((r)=>{console.log('mod = ', r);});
                  ws.call('abs', [-33]).catch((e)=>{console.log('catch abs:',e);}).then((r)=>{console.log('abs = ', r);});

                }else console.log('login error');
      })

    // ws.close();  //  асинхронность! вызывает ошибку
});
```

18. **WebSockets:** rpc-websockets, asinc (parallel), RPC, close.

```
const async  = require('async');
const rpcWSC = WebSocket = require('rpc-websockets').Client

let ws = new rpcWSC('ws://localhost:5000');
let h = (x=ws)=>async.parallel({
                sum: (cb)=>{ ws.call('sum', [5, 3]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                sub: (cb)=>{ ws.call('sub', [5, 3]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                mul: (cb)=>{ ws.call('mul', [5, 3]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                div: (cb)=>{ ws.call('div', [5, 3]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                get: (cb)=>{ ws.call('get', [123],3000).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                mod: (cb)=>{
                        ws.login({login: 'smw', password: '777'})
                        .then((login)=>{
                                if (login) ws.call('mod', [33, 5]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));
                                else cb({message1:'login error'}, null);
                        })
                },
                abs: (cb)=>{
                        ws.login({login: 'smw', password: '777'})
                        .then((login)=>{
                                if (login)  ws.call('abs', [-33]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));
                                else cb({message2:'login error'}, null);
                        })
                }
        },
        (e, r)=>{
                if(e) console.log('e =',e);
                else  console.log('r = ',r);
                ws.close();
        }
);
ws.on('open', h);
```

19. **WebSockets:** rpc-websockets, asinc (waterfall), RPC, close.

```
const async  = require('async');
const rpcWSC = WebSocket = require('rpc-websockets').Client
// |(((8+3)-2)*(-4)/2)%4|
let ws = new rpcWSC('ws://localhost:5000');
let h = ()=>async.waterfall([
                (cb)=>{ ws.call('sum', [8, 3]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                (p,cb)=>{ ws.call('sub', [p, 2]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                (p,cb)=>{ ws.call('mul', [p, -4]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                (p,cb)=>{ ws.call('div', [p, 2]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));},
                (p,cb)=>{
                        ws.login({login: 'smw', password: '777'})
                        .then((login)=>{
                                if (login) ws.call('mod', [p, 4]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));
                                else cb({message1:'login error'}, null);
                        })
                },
                (p,cb)=>{
                        ws.login({login: 'smw', password: '777'})
                        .then((login)=>{
                                if (login)  ws.call('abs', [-p]).catch((e)=>cb(e,null)).then((r)=>cb(null,r));
                                else cb({message2:'login error'}, null);
                        })
                }
        ],
        (e, r)=>{
                if(e) console.log('e =',e);
                else  console.log('r = ',r);
                ws.close();
        }
);
ws.on('open', h);
```

20. **WebSockets:** rpc-websockets, event/subscribe/emit

```
const rpcWSS = require('rpc-websockets').Server

let k = 0;

let server = new rpcWSS({port: 5000,  host: 'localhost'});

server.event('event01');
server.event('event02');
server.event('event03');

setInterval(()=>server.emit('event01',  {n:++k, x:1, y:2}),1000);
setInterval(()=>server.emit('event02',  {n:++k, s:'hello', d:'2019-09-09'}),2000);
setInterval(()=>server.emit('event03' , {n:++k}), 3000);
```

```
const rpcWSC = WebSocket = require('rpc-websockets').Client
let ws = new rpcWSC('ws://localhost:5000');

ws.on('open', ()=>{

    ws.subscribe('event01');
    ws.subscribe('event02');
    ws.subscribe('event03');



    ws.on('event01',(p)=>{console.log('event01:', p);});
    ws.on('event02', (p)=>console.log('event02:', p));
    ws.on('event03', (p)=>console.log('event03:',p));

});
```

21. **WebSockets:** rpc-websockets, notify

```
const rpcWSS = require('rpc-websockets').Server

let server = new rpcWSS({port: 5000,  host: 'localhost'});

server.register('notify1', (params) =>{ console.log('notify1',  params)}).public();
server.register('notify2', (params) =>{ console.log('notify2',  params)}).public();
```

```
const rpcWSC = WebSocket = require('rpc-websockets').Client
let ws = new rpcWSC('ws://localhost:5000');

let k = 0;
ws.on('open', ()=>{
    setInterval(()=>ws.notify('notify1', {n:++k, x:1, y:2}),1000);
    setInterval(()=>ws.notify('notify2', {n:++k, x:1, y:2}),5000);
});
```

WebSockets-сервер: https://habr.com/ru/company/ruvds/blog/424557/

WebSockets-сервер: https://www.npmjs.com/package/WebSocket

WebSockets-сервер: https://www.npmjs.com/package/websocket-without-native

WebSockets-сервер: https://www.npmjs.com/package/ws

WebSockets-сервер: https://learn.javascript.ru/websockets

WebSockets-сервер: https://www.programmableweb.com/news/what-websockets-push-styled-api-and-how-does-it-work/analysis/2017/04/20

nodemon: npm install nodemon -g