```
/*
データサーバ API を呼び出すスクリプト
*/

/*
チャネル操作
*/

function get_channel_list(cb) {
    let req = new XMLHttpRequest();
    req.onreadystatechange = function(){
        if (this.readyState == 4 && this.status == 200) {
            let ml = JSON.parse(this.response, 'utf8');
            cb(ml.channel);
        }
    }
    req.open('GET', '/channel/');
    req.send();
}

function add_channel(channel_id, channel_name, channel_owner, cb) {
    let req = new XMLHttpRequest();
    req.onreadystatechange = function(){
        if (this.readyState == 4 && this.status == 200) {
            let ml = JSON.parse(this.response, 'utf8');
            cb(ml);
        }
    }
    let channel = {"id":channel_id, "name":channel_name, "owner":channel_owner};
    let json = JSON.stringify(channel);
    req.open('POST', '/channel');
    req.setRequestHeader('Content-Type', 'application/json');
    req.send(json);

}

function delete_channel(channel_id, cb) {
    let req = new XMLHttpRequest();
    req.onreadystatechange = function(){
        if (this.readyState == 4 && this.status == 200) {
            let ml = JSON.parse(this.response, 'utf8');
            cb(ml);
        }
    }
    req.open('DELETE', '/channel/' + channel_id);
    req.send();

}


/*
データ操作
*/
function get_influxdb_data(channel_id, measurement, from, to, cb) {
    let req = new XMLHttpRequest();
    req.onreadystatechange = function(){
        if (this.readyState == 4 && this.status == 200) {
            let ml = JSON.parse(this.response, 'utf8');
            cb(ml.data);
```

```
            }
        }
        /*
          検索条件設定
        */
        let param = "";
        if (measurement) {
            param = '&measurement='+measurement;
        }
        let url = '/data/?channel_id=' + channel_id + param
        req.open('GET', url);
        req.send();
    }


```

リスト 1

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>チャネル管理</title>
    <script src="js/mylib.js"></script>
    <link rel="stylesheet" type="text/css" href="css/channel.css">
  </head>
<body>
<h1>My データ・サーバ：チャネル管理</h1>
<br>
<div>

<h2>チャネル一覧</h2>
<div id="channeltable"></div>
<br><br>
<h2>チャネル追加</h2>
<form action="" id="channel_add_form">
  ID：<input type="text" name="id" id="channel_id"/><br>
  チャネル名：<input type="text" name="name" id="channel_name"/><br>
  オーナー：<input type="text" name="owner" id="channel_owner"/><br>
  <input type="button" value="登録" onclick="add_channel_act()"/>
</form>
<br>
<br>
<button onclick="history.back()">戻る</button>

<script>

  function load_channel_table() {
      document.getElementById('channeltable').lastElementChild.remove();
      get_channel_list(function(list){
          let table = create_channel_table(list);
          document.getElementById('channeltable').appendChild(table);

      });
  }

```

```
function add_channel_act() {
    let ch_id = document.getElementById("channel_id").value;
    let ch_name = document.getElementById("channel_name").value;
    let ch_owner = document.getElementById("channel_owner").value;
    console.log("add_channel_act:"+ ch_id + "," + ch_name + "," + ch_owner)
    /* チャネル登録 */
    add_channel(ch_id, ch_name, ch_owner, function(res){
        /* テーブル再表示 */
        load_channel_table();
    });
}

function delete_channel_act(channel_id) {
    console.log("delete_channel_act:" + channel_id)

    /* 削除確認 */
    let msg = channel_id + " チャネルを削除します。¥n よろしいですか？";
    let ret = confirm(msg);
    if (ret == true) {
        /* チャネル削除 */
        delete_channel(channel_id, function(res){
            /* テーブル再表示 */
            load_channel_table();
        });
    }

}

function view_channel_act(channel_id) {
    console.log("viewe_channel_act:" + channel_id)
    /* チャネルデータ表示 */
    window.open("./data.html?channel_id="+channel_id , channel_id);
}

function create_channel_table(list) {
    /* 取得したチャネルでテーブルを作成 */
    let table = document.createElement('table');
    /* ヘッダ */
    let tr = document.createElement('tr');
    let th = document.createElement('th');
    th.textContent = "チャネル ID";
    tr.appendChild(th);
    th = document.createElement('th');
    th.textContent = "チャネル名";
    tr.appendChild(th);
    th = document.createElement('th');
    th.textContent = "所有者";
    tr.appendChild(th);
    th = document.createElement('th');
    th.textContent = "操作";
    tr.appendChild(th);
    table.appendChild(tr);
    /* 行 */
    for (let channel of list) {
        let tr = document.createElement('tr');
        let td = document.createElement('td');
        td.textContent = channel['id'];
        tr.appendChild(td);
            td = document.createElement('td');
```

```
 98             td.textContent = channel['name'];
 99             tr.appendChild(td);
100                td = document.createElement('td');
101             td.textContent = channel['owner'];
102             tr.appendChild(td);
103                td = document.createElement('td');
104                btn = document.createElement("input");
105                btn.setAttribute("type", "button");
106                btn.setAttribute("value", "データ表示");
107                btn.setAttribute("onclick", "view_channel_act('" + channel['id'] + "')");
108             td.appendChild(btn);
109                btn = document.createElement("input");
110                btn.setAttribute("type", "button");
111                btn.setAttribute("value", "削除");
112                btn.setAttribute("onclick", "delete_channel_act('" + channel['id'] + "')");
113             td.appendChild(btn);
114             tr.appendChild(td);
115                table.appendChild(tr);
116         }
117             return table;
118         }
119
120         var channel_list = [];
121         get_channel_list(function(list){
122             let table = create_channel_table(list);
123             document.getElementById('channeltable').appendChild(table);
124
125         });
126
127     </script>
128     </body>
129     </html>
```

リスト 2

```
 1     <!DOCTYPE html>
 2     <html>
 3       <head>
 4         <meta charset="UTF-8">
 5         <title>データ</title>
 6         <script src="js/chart.js"></script>
 7         <script src="js/mylib.js"></script>
 8       </head>
 9     <body>
10     <h1>My データ・サーバ：データ表示</h1>
11     <h2 id="channel_name">>チャネル:<h2>
12     <div class="main" id="main_area"></div>
13     <br>
14     <button onclick="window.close()">閉じる</button>
15
16     <script>
17
18       /*
19         この画面では、
20         ・出力期間は絞らない
21         ・更新間隔は、1 分
22         という仕様で実装する。
23         API 的には細かくデータ取得できるので凝った画面を作ってみてください。
```

```
    */

    var channel_id = null;
    let elm = document.URL.split('?');
    if (elm[1]) {
        let param = elm[1].split('=');
        if (param[0] == "channel_id" && param[1] ) {
            channel_id=param[1];
        }
    }
    console.log(channel_id);

    let cn = document.getElementById("channel_name");
    cn.innerHTML = "チャネル：" + channel_id;


    // 間隔の初期値
    var interval = 60 * 1000;
    var mon_h = null;

    function start_monitor() {
        let dt = new Date();
        console.log("start_monitor", dt);
        mon_h = setInterval(process, interval, interval);
        process(interval);
    }

    function process(offset) {
        console.log("process", offset)
        load_chart(channel_id);
    }

    function load_chart(channel_id) {
        console.log("load_chart:", channel_id);
        /*
            画面仕様としては、
        */
        let measurement = null;
        let from = null;
        let to = null;
        let m = null;
        let f = null;
        let v = null;
        get_influxdb_data(channel_id, null, null, null, function(list){       /* callback of get_influxdb_data */
            console.log("enter get_data callback");
            let m_data = {};
            let fields = {};
            let times = {};
            let times_idx = {}
            let timestamp_p = {}

            for (let rec of list['data']) {
                console.log(rec);
                for (let key in rec) {
                    console.log(key);
                    console.log(rec[key]);
                    if (key == "measurement") {
                        m = rec[key];
                    }
```

```
                else if (key == "timestamp") {
                    ts = rec[key];
                }
                else if (key == "source") {
                    s = rec[key];
                }
                else {
                    f = key;
                    v = rec[key];
                }
            }
            /* measurement */
            if (m_data[m] == null) {
                m_data[m] = {};
                times_idx[m] = -1;
                timestamp_p[m] = "20220101000000";
                times[m] = [];
            }
            let idx = times[m].indexOf(ts);
                if (idx == -1) {
                    times_idx[m]++;
                    idx = times_idx[m];
                    times[m][times_idx[m]] = ts
                }
                /* field */
                fields = m_data[m];
                if (fields[f] == null) {
                    fields[f] = [];
                }
                fields[f][idx] = v
            }
            console.log(times);
            console.log(times_idx);
            console.log(fields);
            console.log(m_data);

            // 表示領域を全部削除
            remove_chart_area();

            for (let m in m_data) {
                // 表示する領域を並べる
                add_chart_area(m);

                // ここで ID から名前に書き換える
                let c = document.getElementById(m+"_caption");
                c.innerHTML = m;

                // チャート表示
                color_list = ["rgb(0,0,255)", "rgb(255,0,0)", "rgb(0,128,0)", "rgb(0,255,255)",
                              "rgb(255,0,255)", "rgb(0,255,0)", "rgb(255,255,0)", "rgb(128,128,0)", "rgb(128,128,128)",
                              "rgb(128,0,128)", "rgb(128,0,0)", "rgb(0,0,128)", "rgb(0,128,128)", "rgb(0,0,0)" ];
                data = [];
                let i = 0;
                for (let key in m_data[m]) {
                    data.push({
                        label: key,
                        fill:false,
                        lineTension:0,
                        data:m_data[m][key],
```

```
142                             borderColor: color_list[i]
143
144                         });
145                         i++;
146                 }
147                 let cv = document.getElementById(m + "_chart");
148                 let mychart = new Chart(cv, {
149                         type: 'line',
150                         data: {
151                             labels:times[m],
152                             datasets:data
153                         }
154                 });
155
156             }
157         });   /* callback of get_influxdb_data*/
158
159     }
160
161
162     // 動的に要素を追加
163     function add_chart_area(monitor_id) {
164             let div_element = document.createElement("div");
165             div_element.setAttribute("class", "monitor");
166             div_element.setAttribute("id", monitor_id + "_area");
167             div_element.innerHTML = '<p class="caption" id="' + monitor_id + '_caption' +'">' + monitor_id + '</p><p class="chart"><canvas id="' + monitor_id +
'_chart"></canvas></p>';
168             let parent_object = document.getElementById("main_area");
169             parent_object.appendChild(div_element);
170     }
171
172     function remove_chart_area() {
173         let main_area = document.getElementById("main_area");
174         while(main_area.firstChild){
175                 main_area.removeChild(main_area.firstChild);
176         }
177     }
178
179     start_monitor();
180     load_chart(channel_id);
181
182
183     </script>
184     </body>
185     </html>
```

リスト３

```
mysql> select * from ds_channel;
+-------+--------------------------+----------+
| id    | name                     | owner    |
+-------+--------------------------+----------+
| CQCH1 | データチャネル1          | tsuchiya |
+-------+--------------------------+----------+
1 row in set (0.00 sec)

mysql>
```

リスト４