

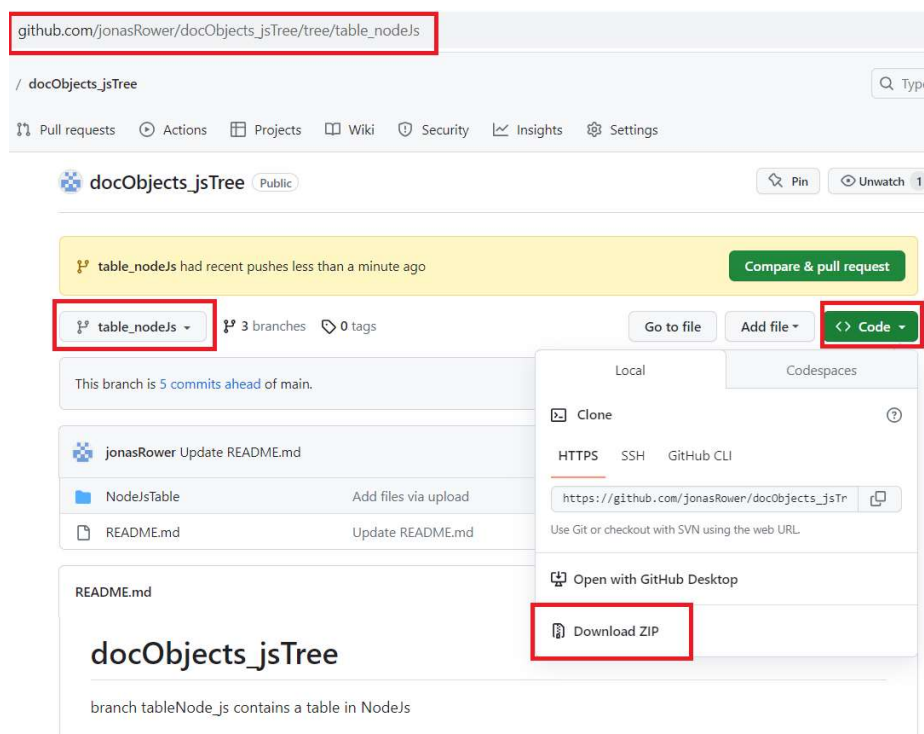
1.	Stažení aplikace.....	2
2.	Instalace knihoven v NodeJs.....	3
3.	Otestování aplikace.....	4
4.	Webové služby aplikace.....	5
5.	Jak aplikace funguje.....	7

Tabulka DocObjects - NodeJs

1. Stažení aplikace

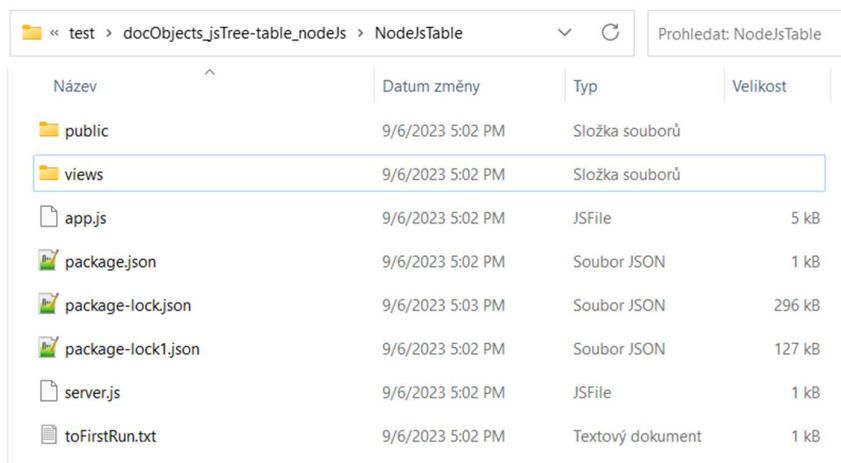
Aplikaci stáhneme zde:

https://github.com/jonasRower/docObjects_jsTree/tree/table_nodeJs



Obr. 1. – Stažení aplikace

Stáhneme a rozbálíme



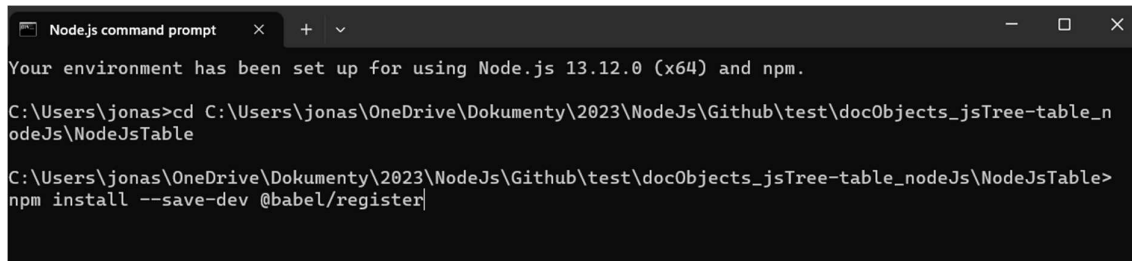
Obr. 2 – Obsah stažené složky

2. Instalace knihoven v NodeJs.

Předpokládáme, že NodeJs máme již nainstalovaný. Stažená složka však neobahuje všechny potřebné knihovny. Ty nainstalujeme takto:

V příkazové řádce vstoupíme do výše uvedené složky a zapíšeme:

```
npm install --save-dev @babel/register
```



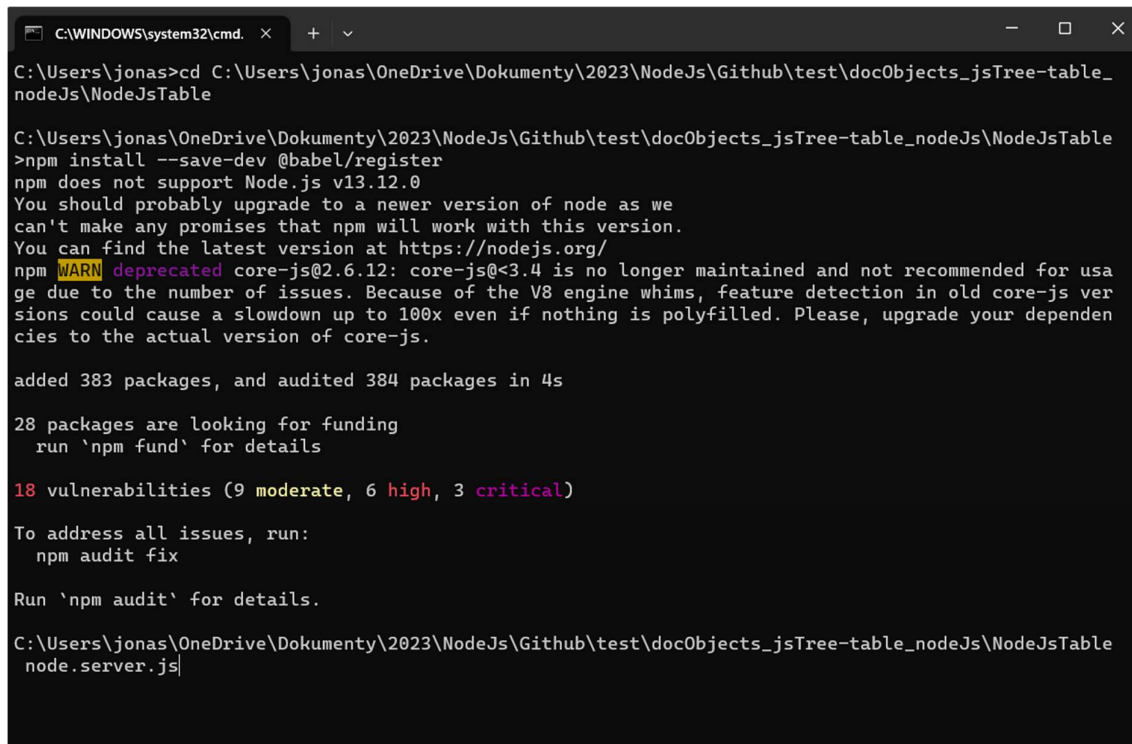
```
Node.js command prompt x + v - □ x
Your environment has been set up for using Node.js 13.12.0 (x64) and npm.

C:\Users\jonas>cd C:\Users\jonas\OneDrive\Dokumenty\2023\NodeJs\Github\test\docObjects_jsTree-table_nodeJs\NodeJsTable

C:\Users\jonas\OneDrive\Dokumenty\2023\NodeJs\Github\test\docObjects_jsTree-table_nodeJs\NodeJsTable>npm install --save-dev @babel/register
```

Obr. 3 – Instalace v cmd

Server spustíme pomocí příkazu `node server.js`



```
C:\WINDOWS\system32\cmd. x + v - □ x
C:\Users\jonas>cd C:\Users\jonas\OneDrive\Dokumenty\2023\NodeJs\Github\test\docObjects_jsTree-table_nodeJs\NodeJsTable

C:\Users\jonas\OneDrive\Dokumenty\2023\NodeJs\Github\test\docObjects_jsTree-table_nodeJs\NodeJsTable>npm install --save-dev @babel/register
npm does not support Node.js v13.12.0
You should probably upgrade to a newer version of node as we
can't make any promises that npm will work with this version.
You can find the latest version at https://nodejs.org/
npm WARN deprecated core-js@2.6.12: core-js@<3.4 is no longer maintained and not recommended for usage
due to the number of issues. Because of the V8 engine whims, feature detection in old core-js versions
could cause a slowdown up to 100x even if nothing is polyfilled. Please, upgrade your dependencies
to the actual version of core-js.

added 383 packages, and audited 384 packages in 4s

28 packages are looking for funding
  run `npm fund` for details

18 vulnerabilities (9 moderate, 6 high, 3 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.

C:\Users\jonas\OneDrive\Dokumenty\2023\NodeJs\Github\test\docObjects_jsTree-table_nodeJs\NodeJsTable>node server.js
```

Obr. 4 – Spuštění serveru

Tabulka DocObjects - NodeJs

Do okna prohlížeče zapíšeme: localhost:3000

localhost:3000

localhost:3000

<<

xx

>>

testA	#A	projectA	1projectA	2projectA	1	2	3	4	5	6
testB	#B	projectB	1projectA	2projectA	1	2	3	4	5	6
testC	#C	projectC	1projectA	2projectA	1	2	3	4	5	6
testD	#D	projectD	1projectA	2projectA	1	2	3	4	5	6
testE	#E	projectE	1projectA	2projectA	1	2	3	4	5	6
testF	#F	projectF	1projectA	2projectA	1	2	3	4	5	6
testG	#G	projectG	1projectA	2projectA	1	2	3	4	5	6
testH	#H	projectH	1projectA	2projectA	1	2	3	4	5	6
testI	#I	projectI	1projectA	2projectA	1	2	3	4	5	6

buttRight

buttLeft



Obr. 5 – Aplikace na localhostu

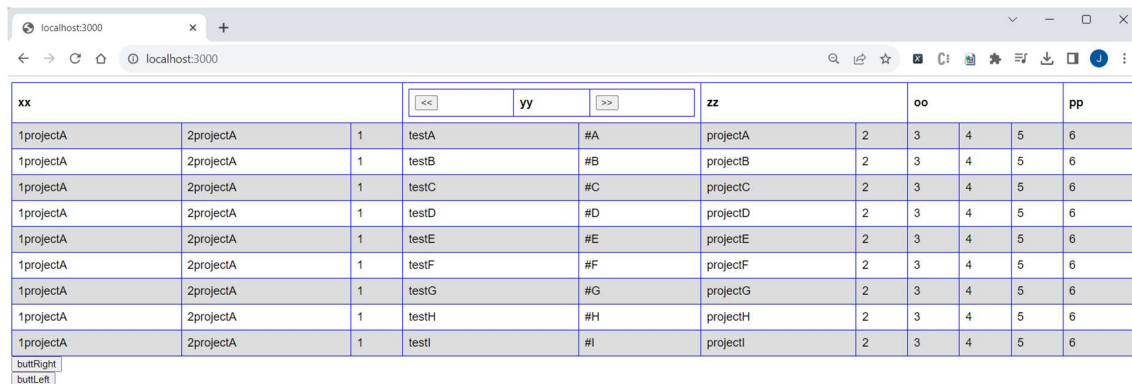
Zobrazí se nám tabulka, kterou jsme již navrhovali zde, avšak čistě v JQuery.



https://github.com/jonasRower/docObjects_jsTree/blob/table_jQuery/doc_Motivace.pdf

(v dokumentaci pro table_JQuery, obr. 5)

3. Otestování aplikace.

Klikáním na šipky   bychom měli posouvat daný sloupec vpravo, či vlevo. Zatím je kód implementován na šipku doprava, doleva nikoliv.



xx			 yy 		zz	oo				pp
1projectA	2projectA	1	testA	#A	projectA	2	3	4	5	6
1projectA	2projectA	1	testB	#B	projectB	2	3	4	5	6
1projectA	2projectA	1	testC	#C	projectC	2	3	4	5	6
1projectA	2projectA	1	testD	#D	projectD	2	3	4	5	6
1projectA	2projectA	1	testE	#E	projectE	2	3	4	5	6
1projectA	2projectA	1	testF	#F	projectF	2	3	4	5	6
1projectA	2projectA	1	testG	#G	projectG	2	3	4	5	6
1projectA	2projectA	1	testH	#H	projectH	2	3	4	5	6
1projectA	2projectA	1	testI	#I	projectI	2	3	4	5	6

Obr. 6 – Zatím je kód implementován na pouze šipku vpravo.

Pod tabulkou se zatím nachází submit tlačítka:



Obr. 7 – Submit tlačítka, budou později skryta. Nachází se vlevo dole, pod tabulkou.

Tabulka DocObjects - NodeJs

Kliknutím na submit tlačítko

`buttRight`

docílíme stejného efektu, jako stisknutím tlačítka

`>>`

4. Webové služby aplikace.

Zapišme do url:

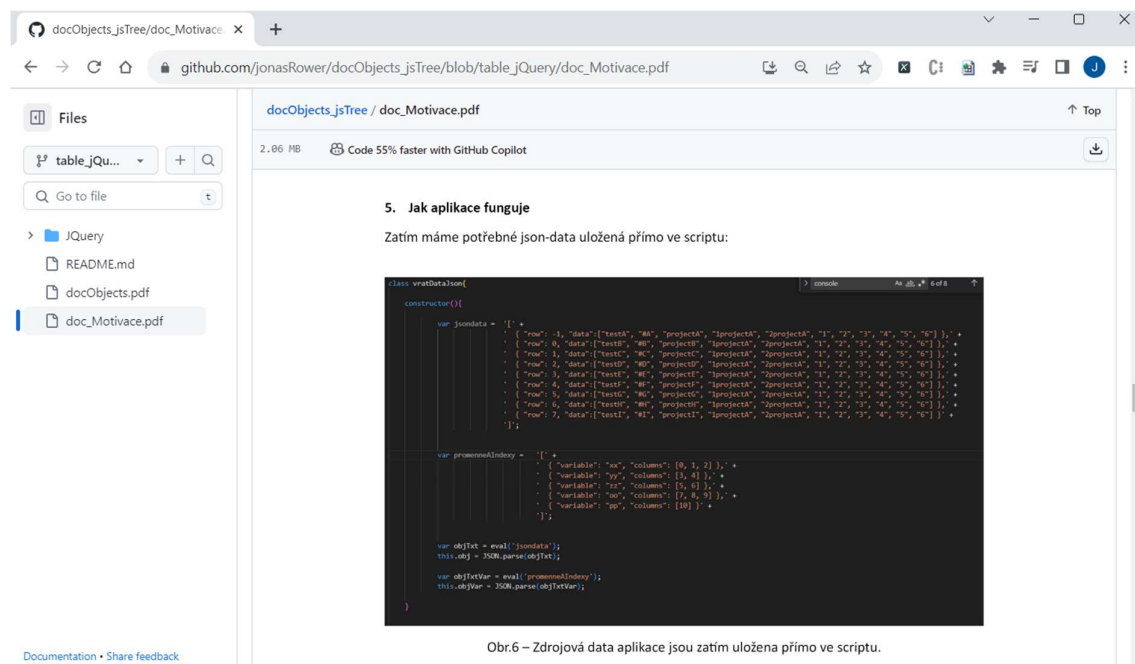
<http://localhost:3000/table/data>



Obr. 8 – Data tabulky.

Data tabulky samozřejmě odpovídají datům, které jsme popsali zde:

https://github.com/jonasRower/docObjects_jsTree/blob/table_jQuery/doc_Motivace.pdf

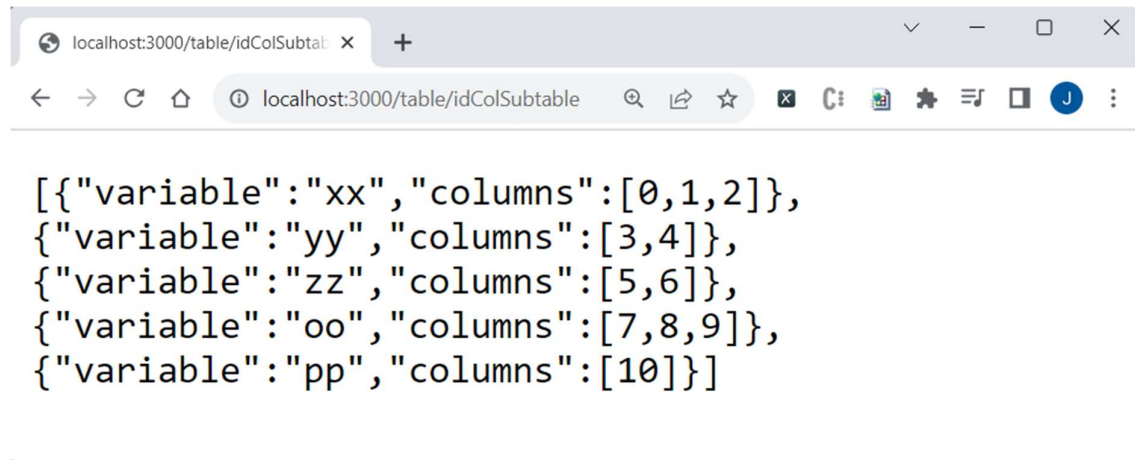


Obr.6 – Zdrojová data aplikace jsou zatím uložena přímo ve scriptu.

Obr. 9 – Zdrojová data v dokumentaci předchozí verze.

Obdobně získáme i:

<http://localhost:3000/table/idColSubtable>

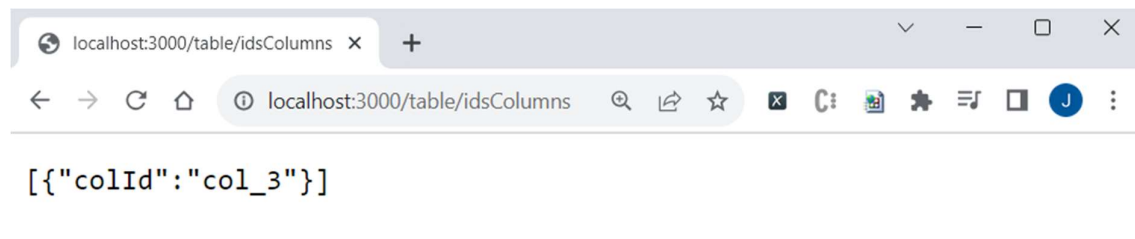


Obr. 10 – Json obsahující indexaci id sloupců a názvů sloupců.

Je tedy evidentní, že sloupce odleva s indexem 0 , 1, 2 jsou sloučené s hlavičkou „xx“. Sloupce s indexem 3, 4 jsou sloučené s hlavičkou „yy“.

Id aktuálního sloupce je zde:

<http://localhost:3000/table/idsColumns>

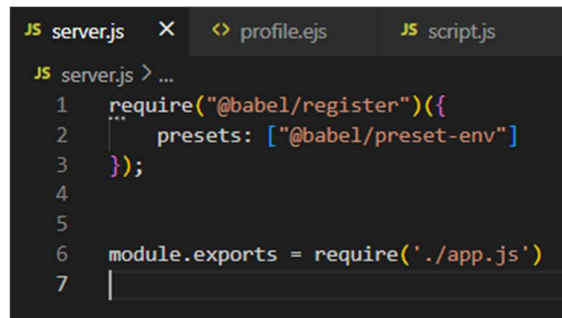


Obr. 11 – V okamžiku volání webové služby zobrazujeme sloupec počínající id=3, tudíž jsme ve sloučených sloupcích s id=3 a 4 pod hlavičkou „yy“.

5. Jak aplikace funguje.

Po spuštění serveru z cmd pomocí příkazu:

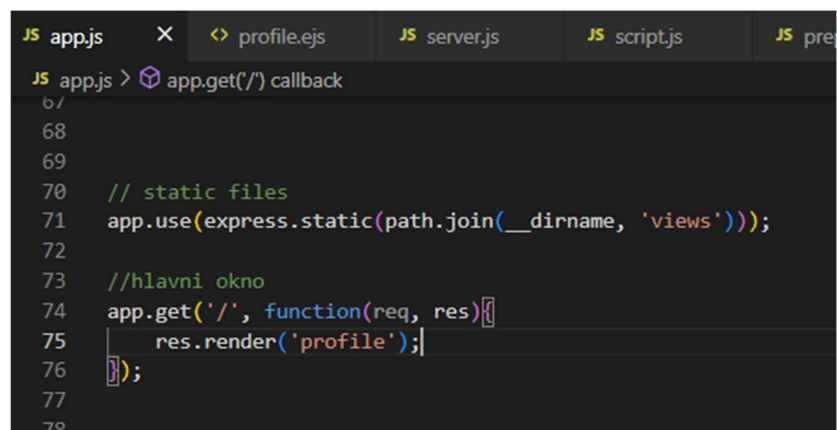
`node server.js`



```
JS server.js X <> profile.ejs JS script.js
JS server.js > ...
1 require("@babel/register")({
2   presets: ["@babel/preset-env"]
3 });
4
5
6 module.exports = require('./app.js')
7
```

Obr. 12 – přesměrování na soubor ./app.js

Jsme přesměrováni do modulu app.js . Zde rendrujeme soubor profile.ejs.



```
JS app.js X <> profile.ejs JS server.js JS script.js JS pre
JS app.js > app.get('/') callback
67
68
69
70 // static files
71 app.use(express.static(path.join(__dirname, 'views')));
72
73 //hlavni okno
74 app.get('/', function(req, res){
75   res.render('profile');
76 });
77
78
```

Obr. 13 – vykreslujeme stránku.

Soubor profile.ejs nalezneme v podsložce views.

```

7 <link href="https://cdnjs.cloudflare.com/ajax/libs/jstree/3.2.1/themes/default/style.min.css" ty
8 <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
9 <script src="https://cdnjs.cloudflare.com/ajax/libs/jstree/3.2.1/jstree.min.js"></script>
10 <script src="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/11.5.1/highlight.min.js"></scri
11
12 <style>
13   table {
14     font-family: arial, sans-serif;
15     border-collapse: collapse;
16     width: 100%;
17   }
18
19   td, th {
20     border: 1px solid blue;
21     text-align: left;
22     padding: 8px;
23   }
24
25   tr:nth-child(even) {
26     background-color: #dddddd;
27   }
28 </style>
29
30 <div id="table"></div>
31
32 <link rel="stylesheet" href="./css/style.css">|
33
34
35
36 <!-- submit na tlačítko doprava -->
37 <form id="nastaveni-form" method="POST" action="/buttRight">
38   <input type="submit" id="subRight" value="buttRight">
39 </form>
40
41
42 <form id="nastaveni-form" method="POST" action="/buttLeft">
43   <input type="submit" id="subLeft" value="buttLeft">
44 </form>
45
46 <script type="text/javascript" src="js/script.js"></script>
47
48
  
```

Obr. 14 – Hlavní stránka projektu.

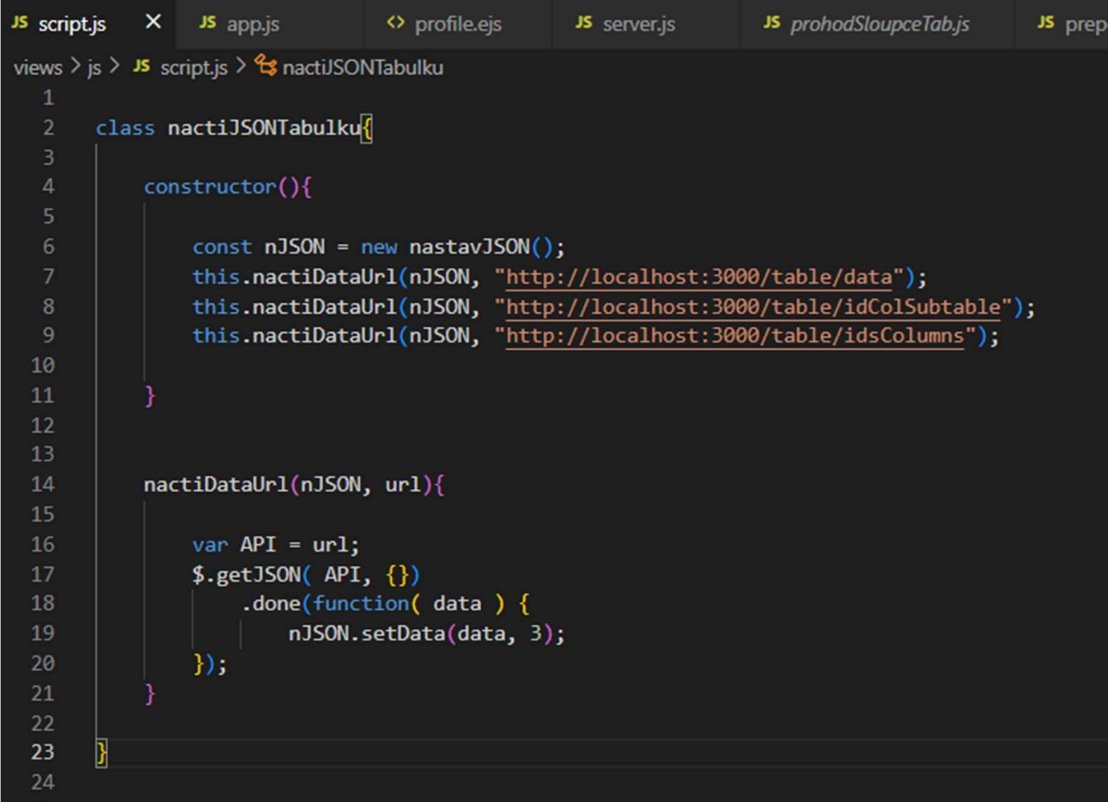
Jedná se prakticky o soubor html. Oproti html, dokáže soubor ejs přijímat parametry přímo z webové služby. Této možnosti zde nevyužíváme, stránku překresluje pomocí js-skriptu, který je zde:

```
<script type="text/javascript" src="js/script.js"></script>
```

Tabulku tedy vykresluje Javascriptem, přidáním stringu pod element:

```
<div id="table"></div>
```


V souboru script.js, přijímáme data z webové služby zde:

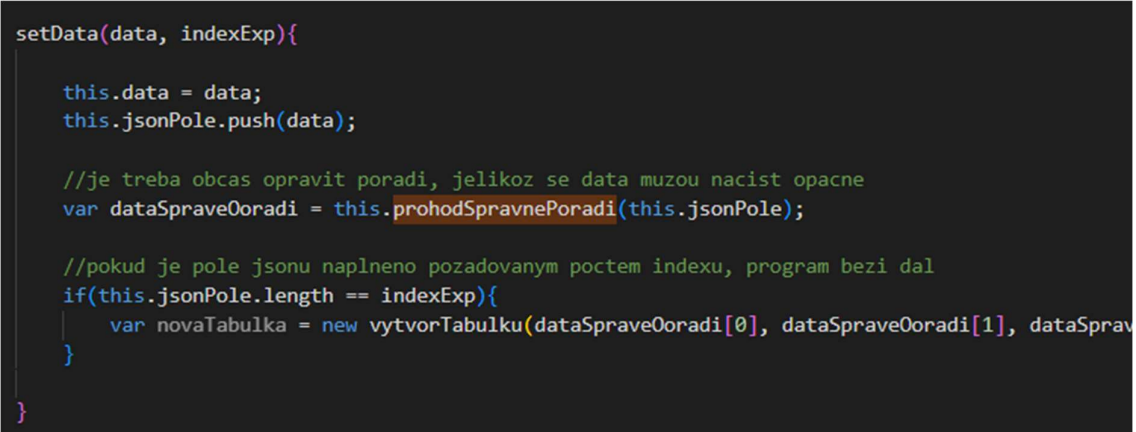


```
1
2 class nactiJSONTabulku{
3
4   constructor(){
5
6     const nJSON = new nastavJSON();
7     this.nactiDataUrl(nJSON, "http://localhost:3000/table/data");
8     this.nactiDataUrl(nJSON, "http://localhost:3000/table/idColSubtable");
9     this.nactiDataUrl(nJSON, "http://localhost:3000/table/idsColumns");
10
11   }
12
13
14   nactiDataUrl(nJSON, url){
15
16     var API = url;
17     $.getJSON( API, {})
18       .done(function( data ) {
19         nJSON.setData(data, 3);
20       });
21   }
22
23 }
24
25
```

Obr. 15 – Načtení dat z webových služeb.

Data předáváme pomocí setru setData do třídy nastavJSON.

Zde je třeba prohodit data z webové služby, jelikož ne vždy se data načtou ve správném pořadí.



```
setData(data, indexExp){
  this.data = data;
  this.jsonPole.push(data);

  //je treba obcas opravit poradi, jelikoz se data muzou nacist opacne
  var dataSpraveOoradi = this.prohodSpravnePoradi(this.jsonPole);

  //pokud je pole jsonu naplneno pozadovany poctem indexu, program bezi dal
  if(this.jsonPole.length == indexExp){
    var novaTabulka = new vytvorTabulku(dataSpraveOoradi[0], dataSpraveOoradi[1], dataSpraveOoradi[2]);
  }
}
```

Obr. 16 – Nastavení dat a jejich prohození, jelikož ne vždy se data z web. Služby načítají ve stejném pořadí.

Následně vytváříme tabulku ve třídě vytvorTabulku.

```
class vytvorTabulku{  
  constructor(obj, objVar, idExpJSON){  
    var idExp = idExpJSON[0].colId;  
    console.log(idExp)  
    var tabAppendStr = this.vytvorTabulkuZJson(obj, objVar, idExp);  
  
    $('#table').append(tabAppendStr);  
  }  
  
  vytvorTabulkuZJson(obj, objVar, idExp){  
    var appendStrHeader = this.vratAppendStrHlavicky(obj, objVar, idExp);  
    var appendStrTable = this.vratAppendStrProTabulku(obj);  
  
    var appendStr = '<table>\n';  
    appendStr = appendStr + appendStrHeader;  
    appendStr = appendStr + appendStrTable;  
    appendStr = appendStr + '</table>\n';  
  
    //console.log(appendStr);  
    return(appendStr);  
  }  
}
```

Obr. 17 – Vytvoření tabulky.

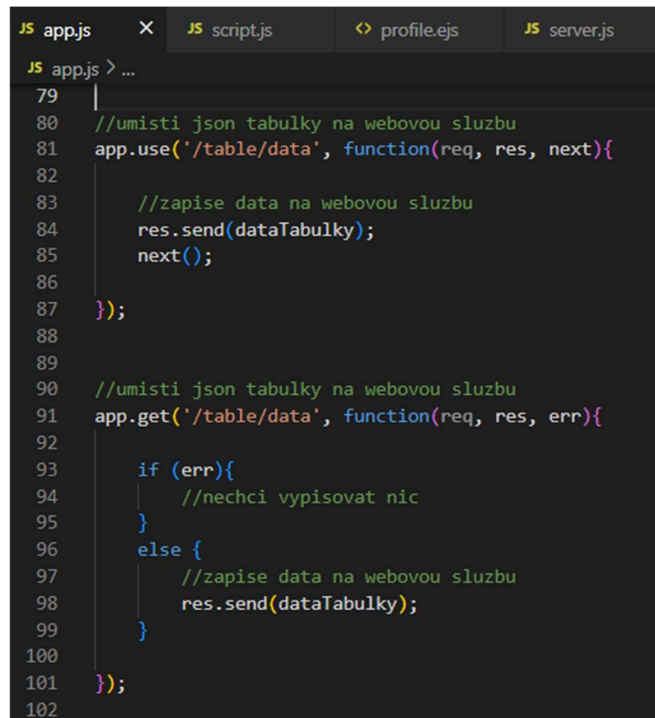
Tabulku vytvoříme pomocí Javascriptu, vytvořením dat v proměnné appendStr, které se přidají pod element table

```
$('#table').append(tabAppendStr);
```

Problematika vytváření tabulky je stejná jako v dokumentaci zde:

https://github.com/jonasRower/docObjects_jsTree/blob/table_jQuery/doc_Motivace.pdf

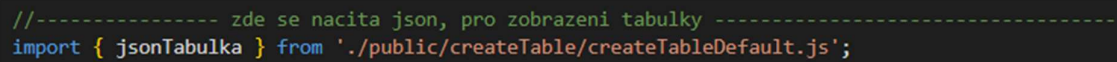
Hlavní data tabulky (Obr. 8), jsme umístili na webovou službu zde – soubor app.js



```
JS app.js X JS script.js <> profile.ejs JS server.js
JS app.js > ...
79 |
80 //umisti json tabulky na webovou sluzbu
81 app.use('/table/data', function(req, res, next){
82
83     //zapise data na webovou sluzbu
84     res.send(dataTabulky);
85     next();
86
87 });
88
89
90 //umisti json tabulky na webovou sluzbu
91 app.get('/table/data', function(req, res, err){
92
93     if (err){
94         //nechci vypisovat nic
95     }
96     else {
97         //zapise data na webovou sluzbu
98         res.send(dataTabulky);
99     }
100
101 });
102
```

Obr. 18 – Umístění jsonu tabulky na webovou službu.

Na Obr. 19 (níže) zobrazujeme získání výchozích dat tabulky. DataTabulky jsou volána z modulu:



```
//----- zde se nacita json, pro zobrazeni tabulky -----
import { jsonTabulka } from './public/createTable/createTableDefault.js';
```

Tabulka DocObjects - NodeJs

```
JS app.js X JS script.js <> profile.ejs JS server.js JS prohodSloupceTab.js JS prepocitejIdsColumns.js
JS app.js > ...
21
22 //----- zde se nacita json, pro zobrazeni tabulky -----
23 import { jsonTabulka } from './public/createTable/createTableDefault.js';
24
25 //----- zde se nacita json, pro ziskani id sloupce -----
26 import { idSloupce } from './public/createTable/idSloupce.js';
27
28 //---- prohazuje sloupce tabulky na zaklade stisknuti tlacitka doprava/doleva -----
29 import { prohodSloupce } from './public/createTable/prohodSloupceTab.js';
30
31 //---- prepocita nove id sloupce, v tom sloupci, ve kterem se bude prekreslovat subtabulka ----
32 import { dopocitejIdsSloupce } from './public/createTable/prepocitejIdsColumns.js';
33
34
35
36
37 //nejake promenne, které se ukladaji na globalni urovni
38 let reqSubmit = ""
39
40
41
42 //nastavi data
43
44 //vychozi data tabulky
45 let dataTabulky;
46
47 //json udavajici indexy sloupce
48 let idSloupceJson;
49
50 //id pro sloupce, kde je subtabulka
51 let aktualniIdSloupce = 0;
52
53
54
55
56 dataTabulky = jsonTabulka();
57 idSloupceJson = idSloupce();
58
```

Obr. 19 – Získání výchozích dat tabulky. Obrázek zachycuje i umístění ostatních modulů, které jsou volány obdobně jako jsonTabulka().

Kde pochopitelně v createTableDefault.js, získáme data do webové služby:

```
JS createTableDefault.js X JS script.js <> profile.ejs JS server.js JS prohodSloupceTab.js JS prepocitejIdsColumns.js
createTable > JS createTableDefault.js > ...

getJsonData(){
    return(this.jsondata);
}

vytvorDefaultniTabulku(){
    var jsondata = '[' +
    ' { "row": -1, "data":["testA", "#A", "projectA", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 0, "data":["testB", "#B", "projectB", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 1, "data":["testC", "#C", "projectC", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 2, "data":["testD", "#D", "projectD", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 3, "data":["testE", "#E", "projectE", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 4, "data":["testF", "#F", "projectF", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 5, "data":["testG", "#G", "projectG", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 6, "data":["testH", "#H", "projectH", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] },' +
    ' { "row": 7, "data":["testI", "#I", "projectI", "1projectA", "2projectA", "1", "2", "3", "4", "5", "6"] }' +
    '];

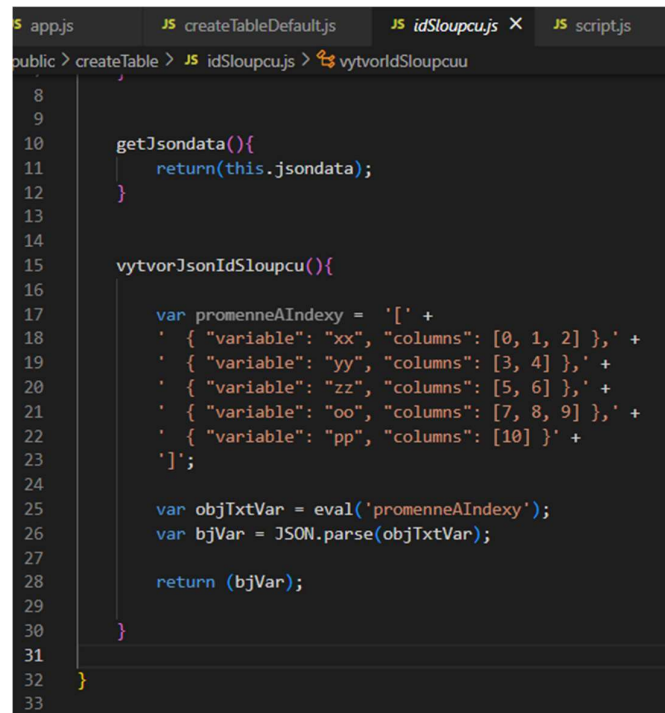
    var objTxt = eval('`'+jsondata+'`');
    var obj = JSON.parse(objTxt);
}
```

Obr. 20 – Výchozí data webové služby.

Obdobným způsobem získáme i data pro webovou službu,

<http://localhost:3000/table/idColSubtable>

viz obrázek 10.

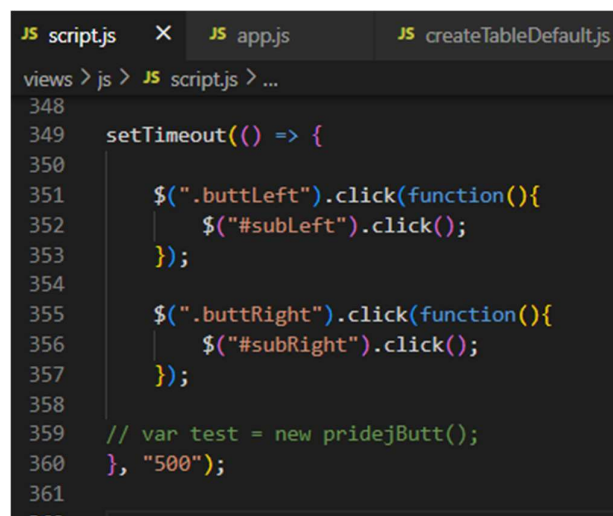


```
8  
9  
10  getJsondata(){  
11      return(this.jsondata);  
12  }  
13  
14  
15  vytvorJsonIdSloupctu(){  
16  
17      var promenneAIndexy = '[' +  
18          '{ "variable": "xx", "columns": [0, 1, 2] },' +  
19          '{ "variable": "yy", "columns": [3, 4] },' +  
20          '{ "variable": "zz", "columns": [5, 6] },' +  
21          '{ "variable": "oo", "columns": [7, 8, 9] },' +  
22          '{ "variable": "pp", "columns": [10] }' +  
23      '];  
24  
25      var objTxtVar = eval('promenneAIndexy');  
26      var bjVar = JSON.parse(objTxtVar);  
27  
28      return (bjVar);  
29  }  
30  
31  }  
32  
33
```

Obr. 21 – Výchozí data webové služby – id sloupců.

Pokud tedy klikneme na tlačítko pak,

v souboru script.js:

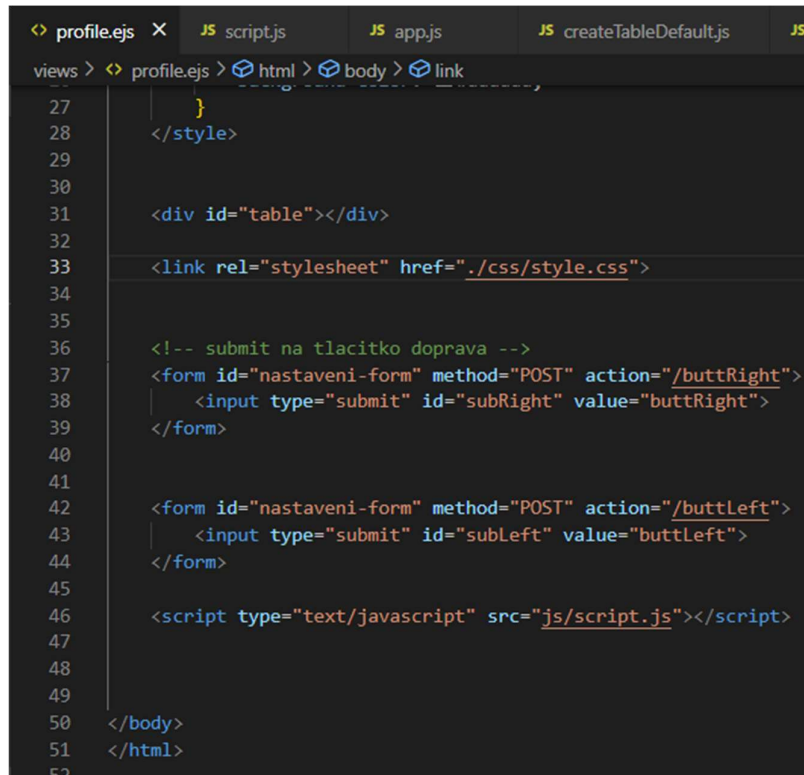


```
JS script.js X JS app.js JS createTableDefault.js  
views > js > JS script.js > ...  
348  
349  setTimeout(() => {  
350  
351      $(".buttLeft").click(function(){  
352          $("#subLeft").click();  
353      });  
354  
355      $(".buttRight").click(function(){  
356          $("#subRight").click();  
357      });  
358  
359      // var test = new pridejButt();  
360      }, "500");  
361  
362
```

Obr. 22 – Funkce sledující stisk tlačítka.


Tabulka DocObjects - NodeJs

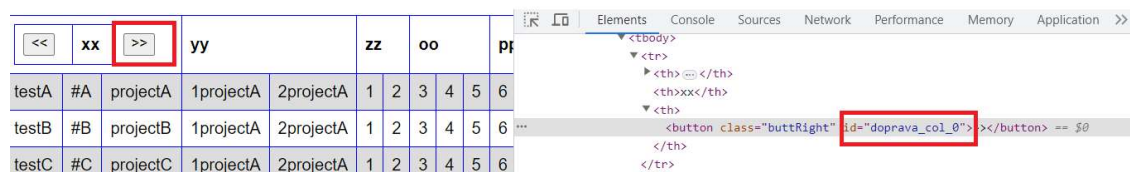
jsou metody, které přeposílají požadavek kliknutí na tlačítko submitu. Aby metody dokázaly reagovat, je třeba nastavit `setTimeout`, jelikož je třeba vyčkat, dokud se stránka nevykreslí. Zde čekáme 500ms.



```
<profile.ejs> X JS script.js JS app.js JS createTableDefault.js JS
views > <profile.ejs> > html > body > link
27   }
28   </style>
29
30
31   <div id="table"></div>
32
33   <link rel="stylesheet" href="./css/style.css">
34
35
36   <!-- submit na tlačítko doprava -->
37   <form id="nastaveni-form" method="POST" action="/buttRight">
38     <input type="submit" id="subRight" value="buttRight">
39   </form>
40
41
42   <form id="nastaveni-form" method="POST" action="/buttLeft">
43     <input type="submit" id="subLeft" value="buttLeft">
44   </form>
45
46   <script type="text/javascript" src="js/script.js"></script>
47
48
49
50 </body>
51 </html>
52
```

Obr. 23 – Submit na straně klienta (profile.ejs).

Tudíž, tlačítko  není submit tlačítko. Jedná se o obyčejné tlačítko, které je generováno pomocí JQuery. Resp. s každým přepnutím do vedlejšího sloupce, se stránka přegeneruje. Tím pádem se vždy vytvoří tlačítko jako nové, s jiným id, avšak se stejným class.

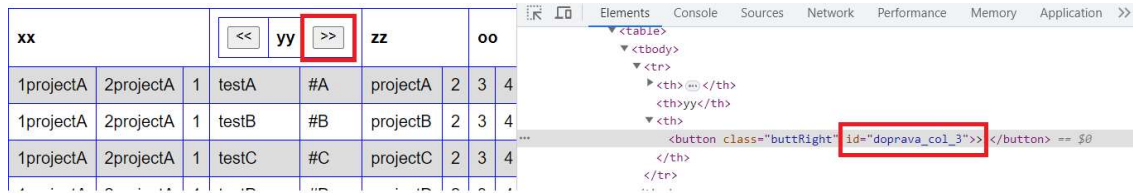


	<<	xx	>>	yy	zz	oo	pt			
testA	#A	projectA	1projectA	2projectA	1	2	3	4	5	6
testB	#B	projectB	1projectA	2projectA	1	2	3	4	5	6
testC	#C	projectC	1projectA	2projectA	1	2	3	4	5	6

Elements Console Sources Network Performance Memory Application >>
<tbody>
 <tr>
 <th><</th>
 <th>xx</th>
 <th>>></th>
 <th>yy</th>
 <th>zz</th>
 <th>oo</th>
 <th>pt</th>
 </tr>
 <tr>
 <td>testA</td>
 <td>#A</td>
 <td>projectA</td>
 <td>1projectA</td>
 <td>2projectA</td>
 <td>1</td>
 <td>2</td>
 <td>3</td>
 <td>4</td>
 <td>5</td>
 <td>6</td>
 </tr>
 <tr>
 <td>testB</td>
 <td>#B</td>
 <td>projectB</td>
 <td>1projectA</td>
 <td>2projectA</td>
 <td>1</td>
 <td>2</td>
 <td>3</td>
 <td>4</td>
 <td>5</td>
 <td>6</td>
 </tr>
 <tr>
 <td>testC</td>
 <td>#C</td>
 <td>projectC</td>
 <td>1projectA</td>
 <td>2projectA</td>
 <td>1</td>
 <td>2</td>
 <td>3</td>
 <td>4</td>
 <td>5</td>
 <td>6</td>
 </tr>
</tbody>

Obr. 24 – Tlačítko vpravo má id="doprava_col_0"

Tabulka DocObjects - NodeJs



Obr. 25 – Tlačítko vpravo má id="doprava_col_3"

Tudíž s každým stiskem tlačítka odesíláme na backend, že jsme dané tlačítko stiskli. V obvyklých případech, bychom spolu s tiskem tlačítka, mohly odesílat i nějaká data (proměnná req).

```

< profile.ejs  JS script.js  JS app.js  X  JS createTableDefault.js  JS idSloupce.js  JS se
JS app.js > app.post('/buttRight') callback
155
156
157 //tlacitko doprava
158 app.post('/buttRight', urlencodedParser, function(req, res){
159
160     var dopravaCol = "doprava_col_" + aktualniIdSloupce;
161     dataTabulky = prohozSloupce(dataTabulky, idSloupceJson, dopravaCol);
162     aktualniIdSloupce = dopocitejIdsSloupce(idSloupceJson, aktualniIdSloupce, 1);
163
164     //po submitu se presmeruje na 'http://localhost:3000/'
165     setTimeout(() => {res.redirect('http://localhost:3000/');}, 600);
166 })
167
168
169 //tlacitko doleva
170 app.post('/buttLeft', urlencodedParser, function(req, res){
171
172     reqSubmit = req.body;
173     console.log(reqSubmit);
174
175     //po submitu se presmeruje na 'http://localhost:3000/'
176     setTimeout(() => {res.redirect('http://localhost:3000/');}, 600);
177 })
178

```

Obr. 26 – Metody post na straně backendu. Můžeme se přesvědčit, že u /buttLeft není implementován kód, tudíž tlačítko zatím nefunguje.

Po přijetí dat na webovou službu ze strany klienta se aktualizuje:

```
var dopravaCol = "doprava_col_" + aktualniIdSloupce;
```

Jelikož aktualniIdSloupce si pamatujeme. Následně aktualizujeme data tabulky:

```
dataTabulky = prohozSloupce(dataTabulky, idSloupceJson, dopravaCol);
```


To provádíme v modulu zde (viz obr. 19):

```
//---- prohazuje sloupce tabulky na zaklade stisknuti tlacitka doprava/doleva -----  
import { prohozSloupce } from './public/createTable/prohodSloupceTab.js';
```

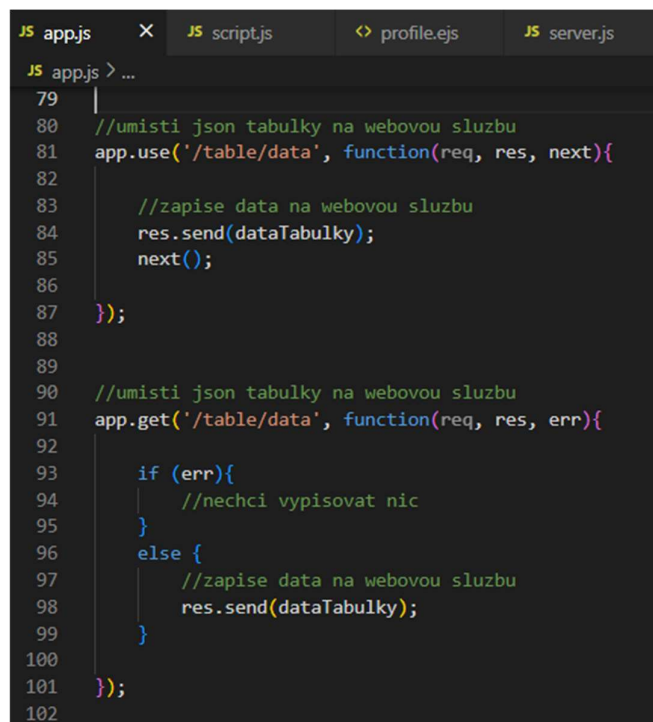
Následně aktualizujeme id sloupce.

```
aktualniIdSloupce = dopocitejIdsSloupce(idSloupceJson, aktualniIdSloupce, 1);
```

To provádíme v modulu zde (viz obr. 19):

```
//---- prepocita nove id sloupce, v tom sloupci, ve kterem se bude prekreslovat subtabulka ----  
import { dopocitejIdsSloupce } from './public/createTable/prepocitejIdsColumns.js';
```

Tudíž opět jsou dostupná nová, aktualizovaná data na webových službách o kterých jsme se již bavili výše.



```
JS app.js  X  JS script.js  <> profile.ejs  JS server.js  
JS app.js > ...  
79 |  
80 | //umisti json tabulky na webovou sluzbu  
81 | app.use('/table/data', function(req, res, next){  
82 |  
83 |     //zapise data na webovou sluzbu  
84 |     res.send(dataTabulky);  
85 |     next();  
86 |  
87 | });  
88 |  
89 |  
90 | //umisti json tabulky na webovou sluzbu  
91 | app.get('/table/data', function(req, res, err){  
92 |  
93 |     if (err){  
94 |         //nechci vypisovat nic  
95 |     }  
96 |     else {  
97 |         //zapise data na webovou sluzbu  
98 |         res.send(dataTabulky);  
99 |     }  
100 |  
101 | });  
102 |
```

Obr. 27 – Cyklicky se vracíme k obrázku 18, odkud se opakuje mechanismus.