TimTable - Ohjeet jatkokehittäjille

Projekti: Titus

Dokumentin laatijat: Matti Leinonen, Ronja Lindholm ja Visa Naukkarinen

Laadittu: 20.5.2018

Muokattu: 26.6.2018

Lisätiedot: `tim@jyu.fi`

Ohjeen tarkoitus ja sisältö

Dokumentti kuvaa Titus-projektin kehittämän timTable-pluginin toimintaa ja rakennetta. TimTable on monipuolinen taulukkomuoto, joka mahdollistaa taulukolle laajat tyyliominaisuudet ja taulukon sisällön edioimisen graafisesti. Tämän dokumentin on tarkoitus antaa pluginin toteutusratkaisuista kokonaiskuva, joka tukee koodin ja sen kommenttien ymmärtämistä.

Tarpeelliset linkit

Tämän dokumentin lisäksi jatkokehittäjälle hyödyllisiä tietoja löytyy myös seuraavien linkkien takaa:

- TimTable käyttöohjeet sisältävät käyttäjän ohjeet sekä sisältö- että taulukkoeditorista. https://tim.jyu.fi/view/kurssit/tie/proj/2018/titus/kayttoohjeet/timtable/yaml.
- Lähdekoodi gitlabissä. Titus-projektin haara sisältää timTable-plugin lisäksi myös muita parannuksia erityisesti tiedekunnan pöytäkirjojen toimintojen helpottamiseksi. https://gitlab.com/Rampastring/tim.

TimTable-pluginin rakenne ja sijainti TIMissä

TimTableen liittyvä koodi sijaitsee TIMissä pääosin tiedostoissa

- `timApp/plugins/timTable.py`,
- `timApp/static/scripts/tim/components/timTable.ts` ja
- `timApp/plugins/timTableLatex.py`.

Tiedostossa `timApp/static/scripts/tim/directives/pareditor.ts` on kappale-editorissa sijaitsevan painikkeen koodi, jolla voidaan luoda yksinkertainen taulukko. Tiedostoissa `timApp/containerLink.py` ja `timApp/tim.py` on pienet lisäykset TimTable-pluginin yhdistämiseksi osaksti TIMiä. Lisäksi tiedostossa `timApp/tests/browser/test_timtable.py` on pluginille osin automatisoituja testejä.

Toiminnon kuvaus

TimTable-plugin käyttää YAML-merkintäkielen pohjalle rakennettua omaa taulukkomuotoa. Taulukkomuoto koostuu tietyistä varatuista sanoista tai termeistä sekä niiden kirjoittamises-

ta YAML:in mukaisesti oikein sisennetysti ja rivitetysti. TimTable-taulukkomuotoa kuvataan käyttäjän ohjeissa `https://tim.jyu.fi/view/kurssit/tie/proj/2018/titus/kayttoohjeet/timtabl`

Taulukon YAML:ia voi muokata kokonaisuudessaan kappale-editoria käyttämällä. Taulukon solujen sisältöä voi muokata myös sisältöeditorilla aktivoimalla ensin taulukon muokkaustilan. Sisältöeditorin käyttö muuttaa YAMLista vain `tabledatablock`-elementtiä. Taulukon solujen sisällöt luetaan aina ensisijaisesti `tabledatablock`ista, jos solun arvo on määritelty siellä. Tällöin alkuperäisen taulukon saa helposti takaisin poistamalla `tabledatablock`in. Rvin ja sarakkeen lisäyspainikkeilla voidaan taulukkoon lisätä rivejä ja sarakkeita. Nämä painikkeet muuttavat taulukosta muitakin osia kuin `tabledatablock`ia.

Aina kun solu avataan taulukon muokkaustilassa, haetaan sen sisältö palvelimelta kutsumalla `timTable.py`-tiedostossa sijaitsevaa reittiä `getCellData`. Näin saadaan muokattavaksi aina solun ajantasainen sisältö ja lisäksi esimerkiksi markdown-sisältö raakatekstinä eikä `HTML`-muodossa.

Muokatun solun sisällön tallentaminen tapahtuu kutsumalla `timTable.py`-tiedostossa sijaitsevaa reittiä `saveCell` Tämä reitti palauttaa solun sisällön `HTML`-muotoon muutettuna, jotta mahdolliset markdown-muotoilut saadaan näkymään selaimessa oikein.

TimTablen reittien tiedosto timTable.py

Tiedostossa `timTable.py` sijaitsevat seuraavat reitit:

- `multihtml/` palauttaa taulukon `HTML`:n.
- `getCellData` palauttaa taulukon yksittäisen solun sisällön.
- `saveCell` tallentaa taulukon yksittäisen solun sisällön.
- `multimd`/palauttaa taulukon LaTeX-käännöksen.
- `addRow` lisää rivin taulukkoon.
- `addColumn` lisää sarakkeen taulukkoon.

Tiedostossa on näille reiteille myös seuraavat apumetodit:

- `is_datablock` kertoo, löytyykö taulukon YAMLista jo `tabledatablock`-elementti.
- `create_datablock` luo YAMLiin `tabledatablock`in tarvittaessa.
- `save_cell` päivittää `tabledatablock`issa olevan solun sisältöä tai lisää sinne kokonaan uuden solun.
- `find_cell` etsii solun sisällön muualta kuin `tabledatablock`ista. Tätä käytetään silloin, kun solun arvoa ei ole määritelty `tabledatablock`issa.
- `find_cell_from_datablock` etsii solun sisältöä `tabledatablock`-elementistä.
- `colnum_to_letters` muuttaa sarakkeen indeksin kirjaimeksi taulukkolaskentaohjelmien tyyliin.

TimTablen AngularJS-komponentti timTable.ts

Tiedostossa `timTable.ts` sijaitsee käyttäjän taulukolle määrittelemien ominaisuuksien HTML-muotoon renderöinnin koodi sekä TimTable-pluginin käyttöliittymän koodi. Tiedostosta löytyy pluginille AngularJS-ohjelmistokehyksen mukainen HTML-malline sekä kontrolleri `TimTableController`. Lisäksi tiedostossa määritellään taulukon rakenne TypeScript-rajapintoina.

Oleellisimmat attribuutit TimTable-kontrollerissa

TimTable-kontrollerin oleellisimmat attribuutit:

- `cellDataMatrix` on kaksiulotteinen merkkijonotaulukko, jossa pidetään yllä taulukon solujen sisältöä HTML-muodossa. Tämä alustetaan kontrollerin onInit-metodissa. Lisäksi `cellDataMatrix` päivitetään aina kun soluja muokataan sisältöeditorilla.
- `data` sisältää käyttäjän määrittelemät taulukon muotoilut.
- `editing` sisältää tiedon siitä, onko taulukon editointitila päällä.
- `editedCellContent`-attribuuttiin haetaan solun avaamisen yhteydessä palvelimelta solun sisältö muokkaamista varten.
- `currentCell` sisältää muokattavana olevan solun sarake- ja rivi-indeksin sekä tiedon siitä, onko solun sisältö auki kappale-editorissa.
- `mouseInTable` sisältää tiedon siitä, sijaitseeko kursori taulukon päällä.

Oleellisimmat metodit TimTable-kontrollerissa

TimTable-kontrollerin oleellisimmat metodit:

- `onInit` alustaa `cellDataMatrix`-attribuutin kutsumalla metodeita `readDataBlockAndSetValuesTo` ja `InitializeCellDataMatrix`. Metodissa alustetaan myös dokumentille kaksi näppäinkuuntelijaa.
- `saveCells` kutsuu `timTable.py`-tiedostossa sijaitsevaa `saveCell`-reittiä yksittäisen solun sisällön tallentamiseksi. Metodissa myös päivitetään vastauksena saatu solun HTML-muotoinen sisältö `cellDataMatrix`-attribuuttiin.
- `getCellData` kutsuu `timTable.py`-tiedostossa sijaitsevaa reittiä `getCellData` yksittäisen solun sisällön saamiseksi palvelimelta.
- `openBigEditor`, `editorOpen` ja `openEditor` avaavat laajemman editorin solun sisällön muokkausta varten. `openEditor` käsittelee vielä muokkauksen peruutuksen ja muokatun sisällön tallentamisen.
- `InitializeCellDataMatrix` alustaa `cellDataMatrix`-attribuutin sisällön `data`-attribuutista jättäen sen `tabledatablock`-osan huomiotta.
- `readDataBlockAndSetValuesToDataCellMatrix` asettaa `cellDataMatrix`-attribuuttiin `data`-attribuutin `tabledatablock`-osan sisällön.
- `getAddress` muuttaa `tabledatablock`-elementin solun sisältöjen koordinaatit nollasta alkaviksi indeksinumeroiksi, esimerkiksi `A1 -> 0,0` ja `B3 -> 1,2`.
- `keyDownPressedTable` kuuntelee näppäimistöä käyttäjän syötteiden varalta. Syötteisiin reagoidaan vain, mikäli kursori on taulukon päällä.
- `handleArrowMovement` kuuntelee nuolinäppäimiä.
- `stylingForCell`, `stylingForColumn`, `stylingForRow` ja `stylingForTable` muuttavat YAMLiin syötetyt tyyliominaisuudet HTML:n mukaisiksi tyyleiksi eri elementeille.
- `addRow` kutsuu tiedostossa `timTable.py` sijaitsevaa reittiä `addRow` uuden rivin lisäämiseksi taulukkoon. Metodi myös päivittää kontrollerin `data`- ja `cellDataMatrix`-attribuutit, jotta muokattu taulukko saadaan näkymään selaimessa oikein.
- `addColumn` kutsuu tiedostossa `timTable.py` sijaitsevaa reittiä `addColumn` uuden sarakkeen lisäämiseksi taulukkoon. Metodi myös päivittää kontrollerin `data`- ja `cellDataMatrix`-attribuutit, jotta muokattu taulukko saadaan näkymään selaimessa oikein.

TimTable-kontrollerin lisähuomiot

Kontrollerin luomassa sisältöeditorissa elementit eivät sijoitu välttämättä tarkoituksenmukaisella tavalla, sillä niiden sijainti on riippuvainen käyttäjän toiminnasta ja taulukon koosta. Plus-ikonit on sijoitettu tyylitiedostossa, joten niiden sijainti on vakaa. Syötekenttä puolestaan sijoitetaan jqueryllä, minkä johdosta sen sijainti on hieman epävakaampi. Havaittuja ongelmia ovat seuraavat:

- Hyvin pienessä taulukossa syötekenttä peittää sarakkeen lisäyspainikkeen.
- Syötekenttä ei sijoitu vasempaan yläkulmaan, jos taulukon koko vaihtuu klikkauksen johdosta. Testeissä on raportoitu syötekentän sijoittumisesta varsinkin solun ylä- ja oikealle puolelle.

LaTeX-tulostus timTableLatex.py

TimTable-pluginin LaTeX-tulostus suoritetaan moduulissa `timTableLatex.py`, joka sijaitsee kansiossa `timApp/plugins/`. Moduuli sisältää kaiken `TimTable`-taulukon LaTeX-taulukoksi muuntamiseen liittyvän toiminnallisuuden lukuun ottamatta osaa solujen sisältöjen muunnoksista, jotka TIM suorittaa toisaalla ennen moduulin käyttöä.

Moduulin rajapintana toimii funktio `convert_table`, joka ottaa syötteenä `TimTable`-taulukon JSON-muodossa ja palauttaa sen LaTeX-muodossa. Moduulia käytettäessä sen muita funktioita ja luokkia ei kutsuta suoraan, vaan `convert_table` tekee kaiken.

LaTeX-tulostuksen luokat

Moduulin `timTableLatex` toiminta perustuu olioihin. LaTeX-tulostuksessa taulukon sisältö luetaan sekä taulukon, rivien ja solujen ominaisuudet tallennetaan vastaaviin olioihin. Moduulin käyttämien olioiden luokkia ovat seuraavat:

- `Cell` sisältää yhden taulukkosolun indeksin taulukkorivillä, sisällön ja muotoilut sekä rajat `CellBorders`-oliossa.
- `CellBorders` sisältää yhden solun jokaisen neljän rajan tiedot mukaan lukien, piirretäänkö ne ja minkä värisiä ne ovat.
- `Row` vastaa yhtä taulukon riviä sisältäen listan `Cell`-soluolioista sekä seuraavat metodit:
  - `add_cell` tekee solujen lisäyksen tiettyyn indeksiin, mitä käytetään monirivisten solujen muodostamiseen LaTeXilla.
  - `get_colspan` palauttaa rivin solujen yhteenlasketun sarakemäärän, joka poikkeaa solujen määrästä, jos rivissä on monisarakkeisia soluja.
  - `get_row_height` palauttaa rivin korkeimman solun korkeuden, joka asetetaan rivin korkeudeksi, sillä muunnos ei tue eri korkuisia soluja.
- `HorizontalBorder` vastaa rivienvälisten viivojen piirtämisestä sisältäen viitteet viivan ylä- ja alapuolisiin riveihin.
- `Table` sisältää listat yhden taulukon kaikista `Row`- ja `HorizontalBorder`-olioista sekä seuraavat metodit:
  - `create_hborders` luo `HorizontalBorder`-oliot taulukon rivien perusteella.
  - `get_or_create_row` joko palauttaa indeksiä vastaavan rivin tai luo uuden, jos sitä ei ole vielä olemassa.
- `TimTableException` sekä sen perivät `ColorDefinitionsMissingError`, `IndexConversionError` ja `TableBorderException` ovat moduulin omia virheluokkia.

Näistä luokat `Cell`, `Row`, `HorizontalBorder` ja `Table` sisältävät lisäksi mukautetun `__str__`-metodin (eli olion merkkijonoksi muuttavan metodin), jolla muutos LaTeX-koodiksi tehdään. Metodissa olion sisältö ja attribuutit luetaan sekä lisätään LaTeX-merkintöjen arvoiksi. Sama toistetaan rekursiivisesti mahdollisille olion listaamille olioille. Esimerkiksi `Table`-olion `__str__`-metodi sekä lisää taulukon asetukset niitä vastaaviin LaTeX-merkintöihin että pyytää `Row`- ja `HorizontalBorder`-olioiden `__str__`-metodeilta niiden täydelliset LaTeX-muodot. Ne edelleen perustuvat `Row`-olioissa listattujen `Cell`-olioden `__str__`-metodien palauttamiin solutason muotoiluihin.

LaTeX-tulostuksen funktiot ja oletusarvot

Rajapintafunktion `convert_table` ja luokkien metodien lisäksi `timTableLatex.py` sisältää apu- ja muunnosfunktioita, joilla attribuutit saadan JSON-datasta. Näitä ovat muun muassa seuraavat:

- `get_color` hakee ja muotoilee taulukon, rivin tai solun teksti- tai taustavärin palauttaen värin sekä tunnisteen siitä, onko väri nimi vai heksakoodi. Jos väriattribuuttia ei ole asetettu, funktio palauttaa sille parametrina annetun oletusarvon.
- `update_content_from_datablock` päivittää taulukon muiden tietojen haun jälkeen datalohkosta mahdolliset muutokset solujen sisältöihin.
- `get_borders` lukee taulukon, rivin tai solun datasta reunaviivojen tiedot ja luo vastaavan `CellBorders`-olion.

Useimmat näistä apufunktioista hakevat JSON-datasta avainta vastaavan arvon ja palauttavat oletuksen, jos avainta ei löytynyt. Oletuksena käytetään joko kovakoodattua perusarvoa tai käyttäjän taulukon ylemmälle tasolle syöttämää asetusta. Esimerkiksi jos rivitasolla tekstin fonttikooksi on asetettu `25`, jokaiseen rivin soluun tulee oletuksena sama fonttikoko `25`, ellei solulle ole asetettu poikkeavaa fonttikokoa.

Jos muotoilulle ei ole asetettu samalla tai ylemmällä tasolla arvoa, sen arvoksi tulee moduulin laajuinen oletusarvo. Esimerkiksi jos fonttikokoa ei ole asetettu, solujen fonttien kooksi tulee tällöin moduulin oletusarvo `10`. Kaikki asetuksien oletusarvot on listattu moduulin alussa.

LaTeX-merkinnät

LaTeX-tulostuksessa TimTable-taulukko muunnetaan LaTeX-muotoiluiksi. Näin ollen esimerkiksi yksinkertainen yhden solun taulukko (ks. kuva 1), jonka YAML-muoto on

```
table:
    rows:
      - row:
        - cell: "Esimerkkisolu"
          backgroundColor: '#123456'
          color: white
          fontFamily: Times
          border: 1px solid red
```

tulostuu seuraavanlaiseksi LaTeX-koodiksi:

```
\begin{table}[h]
\begin{tabular}{c}

\hhline{>{\arrayrulecolor{red}}-}
```

```
\multicolumn{1}{!{\color{red}\vrule}l!{\color{red}\vrule}}{\multirow{1}{*}{
\cellcolor[HTML]{123456}\fontsize{10}{0}\selectfont{\textcolor{white}{{\fontfamily{ptm}\selec
\centering Esimerkkisolu}}}}}}

\tabularnewline[0pt]
\hhline{>{\arrayrulecolor{red}}-}

\end{tabular}
\end{table}
```

Kuva 1: Kuva 1. Yhden solun taulukko.

Esimerkin LaTeX-koodi kattaa useimmat TimTable-taulukoiden LaTeX-tulostuksessa käytettävät muotoilut. Niiden merkitykset ovat seuraavat:

- `\begin{table}[h]` aloittaa taulukon ja `\end{table}` päättää sen. Lisämuotoilu `[h]` pakottaa taulukon asettumaan sivulla samaan kohtaan kuin se on tekstin seassa. Ilman sitä taulukot siirtyvät sopiviin riittävästi tilaa sisältäviin kohtiin.
- `\begin{tabular}{c}` on osa taulukon aloittamista ja vastaavasti `\end{tabular}` sen päättämistä. `c` käytännössä merkitsee sarakkeiden määrää taulukossa. Jos sarakkeita on esimerkiksi kolme, hakasulkujen välissä olisi tällöin `ccc`. Yksinkertaisemmissa taulukoissa kukin `c` lisäksi määrittäisi tekstien keskittämisen osoittamassaan sarakkeessa, mutta taulukon muut merkinnät ylikirjoittavat tämän, joten kirjaimella ei ole vaikutusta asetteluun.
- `\hhline{>{\arrayrulecolor{red}}-}` on taulukon vaakaviiva. Tavuviiva `-` merkitsee viivaa ja `>{\arrayrulecolor{red}}` asettaa sen värin. Esimerkissä viivoja on vain yksi, mutta suuremmissa taulukoissa viivoja voi olla peräkkäin useampia. Jos viivaa ei piirretä, tyhjää väliä merkitään matomerkillä `~`.
- `\multicolumn{1}` aloittaa solun käyttäen monisarakemerkintää, jossa numero `1` merkitsee sarakkeiden määrää. Esimerkin solu kattaa vain yhden sarakkeen.
- `{!{\color{red}\vrule}l!{\color{red}\vrule}}` sisältää tekstin vaakasuoran asettelun vasemmalle (`l`) sekä pystysuorat reunaviivat (`!{\color{red}\vrule}`) vasemmalla ja oikealla värin kera. Tekstin asettelu voi olla myös oikealle eli `r` sekä keskitetty eli `c`. Jos reunaviivaa ei piirretä jommalle kummalle tai molemmille puolille, vastaavat merkinnät jätetään pos.
- `{\multirow{1}{*}{` sisältää solun sisältämien rivien määrän ja solun leveyden. Esimerkissä solu sisältää vain yhden rivin, mutta esimerkiksi kolmen rivin korkuiselle solulle merkittäisiin `-3` (alhaalta ylös päin johtuen taustavärien toimintalogiikasta). Asetus `*` tarkoittaa automaatista solun leveyden asettamista.
- `\cellcolor[HTML]{123456}` määrittää solun taustavärin. Jos solulle ei ole asetettu taustaväriä, asetus jätetään kokonaan pois. Lisäasetus `[HTML]` kertoo, että väri on esitetty heksaluvulla. Jos väri on annettu nimeä (kuten `blue`) käyttäen, lisäasetus jätetään pois.
- `\fontsize{10}{0}\selectfont{\textcolor{white}{{\fontfamily{ptm}\selectfont{` sisältää fontin muotoiluja, kuten sen koon `10`, tekstin värin `white` ja fonttiperheen koodin `ptm`. Jos näitä ei ole erikseen asetettu, niissä käytetään oletusarvoja.
- `\centering` mahdollistaa nimestään huolimatta solun tekstien automaattisen jakamisen useammalle riville solun sisällä, jos solun leveys ei riitä tekstille.
- `Esimerkkisolu}}}}}}` sisältää solun tietosisällön sekä muiden muotoilujen päättämis-

sulkeet. Erilaiset sisällöt, kuten tekstimuotoilut ja matemaattiset kaavat tulevat myös tähän, mutta useimmissa tapauksissa TIM tekee tarvittavat muutokset automaattisesti. Jos taulukon rivillä on useampia soluja, ne tulevat tämän jälkeen `&`-merkillä erotettuina.
- `\tabularnewline[0pt] \hhline{>{\arrayrulecolor{red}}-}` määrittää rivinvaihdon sekä reunaviivat. Jos taulukossa on useampia rivejä, ne tulevat vastaavasti tämän jälkeen. Rivinvaihdon lisäasetus `[0pt]` tarkoittaa rivin korkeutta. Rivin minimikorkeus tekstirivin korkeus, joten vaikka arvo on nolla, rivi näkyy silti.

LaTeX-tulostuksen lisähuomiot

Moduulin tarjoama LaTeX-muunnos ei useimmissa tapauksissa tee yksi-yhteen alkuperäistä vastaavaa taulukkoa. Ensisijainen syy tähän on LaTeX-koodin erot HTML-koodiin toimintalogiikan ja ulkoasun osalta. Keskeisimpiä eroja ja niistä johtuvia toteutusratkaisuja ovat seuraavat:

- HTML-taulukoissa muut solut piirretään automaattisesti pois monirivisten solujen (eli solujen, joiden `rowspan` suurempi kuin `1`) tieltä, kun taas LaTeX-taulukoissa ne piirretään oletuksena päällekkäin. Tämän vuoksi LaTeX-muunnoksessa monisarakkeisten solujen (eli solujen, joiden `colspan` on suurempi kuin `1`) piirtämiseksi joudutaan luomaan vastaava määrä tyhjiä soluja monirivisen solun viemään tilaan.
- LaTeXissa monirivisen solun taustaväri voi osittain peittää tekstin, minkä välttämiseksi monirivisyys tulee vaihtaa alhaalta ylöspäin tarkastelluksi eli käyttää negatiivista `rowspan`-arvoa.
- Solujen indeksöinti poikkeaa HTML-taulukosta yllä mainittujen muutoksien vuoksi, mikä tekee reunaviivojen ja sarakkeiden tunnistamisesta monimutkaista.
- HTML-taulukoissa vaakasuorat ja pystysuorat reunaviivat tulevat suoraan elementin muotoiluista, mutta LaTeX-taulukoissa vaakasuorat reunat ovat irrallisia elementtejä, mikä vaikeuttaa niiden sovittamista oikeille paikoille.
- LaTeXissa vaakasuoria viivoja voi olla vain yksi per riviväli. Esimerkiksi, jos yläpuolisella solulla on erivärinen reunaviiva kuin alapuolisella solulla, vain toista voidaan noudattaa. `timTableLatex`-moduulissa yläpuolinen muotoilu ottaa etusijan.
- Reunaviivojen paksuus ja tyylimuotoilut (kuten katkoviiva) eivät ole tuettuja LaTeX:issa, joten kaikki tulostuksen reunaviivat ovat yhtenäisiä yhden pikselin levyisiä viivoja.
- LaTeX-taulukon soluilla tekstin pystysuoralle tasaukselle ei ole tiedossa toteutusratkaisua, joka olisi yhteensopiva muiden muotoilujen kanssa.
- LaTeX-taulukko ei tue sellaisenaan *display format* -kaavoja (eli kaavoja, jotka käyttävät kahta `$`-merkkiä). Näitä kaavoja sisältävät solut käyttävät `minipage`-muotoilua, jonka puutteena on, että taulukko ei tunnista sen sisällön kokoa automaattisesti. Kaavojen näyttämiseksi oikein näissä tapauksissa käyttäjä joutuu lisäämään solun korkeuden ja leveyden manuaalisesti.

Kehitysideat

Tähän lukuun on kirjattu ylös Titus-projektin aikana esiin tulleita jatkokehitysideoita TimTable-pluginille.

Sisältöeditori

Sisältöeditoria koskevat jatkokehitysideat:

- Rivin poistopainike viimeisen rivin ja sen tietojen poistamiseksi taulukosta.
- Sarakkeen poistopainike viimeisen sarakkeen ja sen tiedojen poistamiseksi taulukosta. Poistopainikkeiden sijainti on mietittävä tarkasti, jotta vältytään mahdollisilta virhepainalluksilta.
- Lukutilan paluupainike editointitilan sulkemiseksi. Painike voisi olla ruksi-ikoni oikeassa yläkulmassa.
- Mahdollisuus määritellä solu muokattavaksi tai ei muokattavaksi.

Taulukkoeditori

Taulukkoeditoria koskevat jatkokehitysideat:

- Sarakkeille voisi lisätä monipuolisemmat tyyliasetukset. Nykyinen muoto noudattaa HTML5:sta, mutta esimerkiksi sarakkeelle sillä voidaan asettaa vain kolme tyyliasetusta: taustaväri, leveys ja reunat. Muutkin sarakeasetukset voisivat kuitenkin olla käyttäjälle tarpeellisia. Tällöin niitä joudutaan asettamaan manuaalisesti jokaiselle solulle yksitellen.
- Solun tyypeiksi voisi lisätä listan ja matemaattisen kaavan. Tyyppi asetetaan `type`-elementille.
- Solun tyypiksi tulisi voida asettaa matemaattinen kaava, jolloin solun sisältöä tulisi käsitellä Excel-taulukon tyylisesti. Tällöin voisi olla mielekästä käyttää myös jonkinlaisia laskuoperaattoreita tekstissä, kuten Excelistä tutut `=()` -merkit. Matemaattiset kaavat voisi toteuttaa solu- ja saraketasolla.