Brainy Data Newsletter

You are receiving this Newsletter because you expressed an interest in our software or are currently licensing one or more of our software components. This newsletter provides up-to-date information on the progress of our current developments, new software releases, and details about future developments.

Please also visit our news page for a summary of developments over the past twelve months.

https://home.brainydata.com/news.htm

NOTE: NEW SUBSCRIBERS will automatically receive a copy of the last newsletter that we circulated and consequently the date shown at the top of the newsletter may not be current.

Software releases and patches

OWrite version 4.1.3.0 release for OWrite and JS-OWrite is available for download from the pages below.

OWrite Desktop License: https://support.brainydata.com/owrite_su.htm

OWrite JS-Server License: https://support.brainydata.com/jsowrite_su.htm

Desktop/Server:

This release resolves intermittent crashes introduced by 4.1 as well as some other important issues. If you are currently using OWrite version 4.1.x, we recommend you upgrade to 4.1.3.0 as soon as possible.

JS-Client:

We added a new feature that allows the placing of multiple OWrite objects within the same browser window. This enhancement was sponsored by one of our customers to whom we would like to extend our thanks. See our sponsors page for more details https://home.brainydata.com/sponsors.htm.

Ongoing development

We are currently working on OWrite/JS-OWrite version 4.2.0.0 which will include a new field view for JS-Client as well as other improvements for OWrite Desktop.

Our ongoing project to port OCal to the JS-Client, has been delayed slightly due to the important work we carried out for OWrite version 4.1.3.0. We initially aimed at releasing an early alpha by Christmas, but this may now have to wait until after the holidays.

Software Sponsorship

Our software-sponsorship program has existed since the formation of Brainy Data, but we have never vigorously promoted it or fully explained its purpose. This newsletter article aims to at least correct the latter. The software-sponsorship program has financed the development of most of our software such as OWrite, PDFDevice, OCal, OGantt, PDFWriter and various other major and minor software projects. All of our software exists thanks to Omnis developers who have sponsored their development, for which we charge a cost-only fee covering expenses and salaries. Sales and subsequent maintenance and support fees of these products have never been sufficient to cover the costs of their original development in addition to the required continued maintenance thereafter. Our maintenance program has to ensure compatibility with new versions of Studio, new operating systems, as well as withstand the many uses developers throw at our software, consequently very little funds remain to cover new developments.

With the requirement to port our software to JS-Client, the software sponsorship program has become more important than ever. Although our current maintenance and technical support subscriptions provide a steady revenue stream to maintain the existing products, it does not provide sufficient finance to expand our workforce in support of these new long-term projects. Because our maintenance and technical support commitments take priority over new developments, these new larger projects frequently experience delays.

Thus, the sponsorship program has two purposes. Firstly, it provides the means to facilitate developer's special requests for the enhancement of our existing software components or to develop new software components, either of which will benefit in particular the sponsor. Sponsors have direct technical input and are able to negotiate completion dates. Secondly, this service is vital in generating the additional revenue required if we are to expand our workforce. By charging a cost-only fee for special requests, it will facilitate us to hire in additional help. By expanding our workforce, we will be better able to increase our commitment to long-term projects, which in turn secures the long-term future of our services. Without this vital element of sponsorship bigger projects will continue to suffer delays or will not be financially viable.

EurOmnis 2018 - session updates

[a] Writing Javascript Controls

During this year's 'Writing JS-Controls' session, some problems were encountered with event-handling while using the Studio 9 JSON Control Editor. Unfortunately, Michael was not able to resolve it to a satisfactory end during the session, although he hopes that he has provided sufficient insight into what's required when writing your own controls. The problem has now been narrowed down as follows: When Omnis produces the javascript for client methods, custom event constants are correctly encoded if used within the $event method, i.e. custom event constants are encoded as strings and built-in event constants as integers. But when used in other methods (for example the class method of the remote form), the javascript that is produced does not encode the custom event constants correctly. In this scenario, all custom event constants were encoded as integer value zero. As far as Michael is aware, this applies to Studio 9 beta as well as Studio 8.1.6 and previous versions of Studio.

This issue has been reported (Reference: ST/EM/214) and we have been told that this has been fixed. In the meantime, the work-around is to specify the names of the custom events as strings within client methods other than $event methods.

Michael will upload the new example library with the work-around to Github, as soon as he can.

[b] Writing External Components C++

During the C++ session, an attempt was made to build David McKeone's non-visual component template, which is publicly available on GitHub. Although most everyone succeeded in updating the template projects for their specific versions of XCode or Visual Studio, it took longer than we would have liked to get the component to build and run. This had various reasons related to unfamiliar XCode versions and Windows Security. Also the code for the worker object, that was outside the scope of this session, took some time and effort to exclude from the builds.

In the end there was little time left to play. Michael just about managed to demonstrate how to write an external notation method that defines a list from an SQL class and populates that list with arbitrary data. How to manipulate list data is an important skill when writing external components. It demonstrated how to query the data types of variables or list-columns and what functions to use to manipulate the different types of data.

What remains to be done by Michael is to tidy up the resulting Macintosh and Windows projects and perhaps also make them available on Github together with a sample library. He will do this in the new year, as he has very little time left before the XMas Holidays.

[c] Commercial Session - JS-OWrite

During the commercial session, Michael and Perry discussed the reason for the approach they took in developing JS-OWrite. Michael and Perry pointed out that all of the competitors they had investigated, suffered various compatibility issues involving font wrapping and other limitations (such as no effective header and footer support or proper visual pagination), resulting in no true WYSIWYG experience by any of these products. This discussion was prompted by the JS-OWrite work they carried out earlier in the year which resulted in large improvements in the WYSIWYG performance of JS-OWrite. However, during this discussion, some of the audience asked if we had investigated the Microsoft Web Office (also called Microsoft Web Apps). Michael and Perry had to admit that we had not. Michael has now rectified this.

During a thorough investigation of the Microsoft Javascript version of Word to discover what we may learn from it, Michael uncovered various short-comings within the JS version of Word. The following are just some of the problems he found. Text-wrapping around images was limited to left and right alignment of images (i.e. text would wrap only on one side of the image). Text-wrapping around images was further limited to a single paragraph. When one tried to add a second paragraph next to the image (by pressing return), the new paragraph was in fact placed below the image creating a large gap between the two paragraphs. When producing a PDF file from a Word Document, the PDF did not at all represent what the Web App was showing. Michael also found inconsistent word-wrapping when the document just contained text. In our opinion, the Word Web App offered a very poor WYSIWYG experience.

We are therefore very proud of our achievement with JS-OWrite. Not only does JS-OWrite perform superbly in regards to WYSIWYG, it also supports all the same word-processing features that the desktop counterpart supports, such as comprehensive header and footer support, tables, sophisticated image-wrapping, fields for merging data and more. More importantly, with the extra features that Michael and Perry implemented earlier this year, output by PDFDevice and JS-OWrite match. We feel JS-OWrite is an outstanding achievement already and it will only get better from hereon.

Technical Hint: JS Browser Caching

The 'Technical Hint' is a new article that we intend to include in all our future Newsletters. It will contain one or more technical hints, things that may have come up repeatedly in technical support queries or we encountered during development.

This article addresses the problem of Browser caching in relation to JavaScript files. While developing JS-OWrite we became aware that many browsers do not automatically refresh javascript files (even though they have changed on the server), especially when the HTML file that references them has not changed. Clicking the refresh button does not solve the problem, but clearing the cache does. However, clearing the cache is not an acceptable solution for end-users. After some research we did discover a solution (thanks to stackoverflow.com). The trick is to add a parameter to the line in the html that references the javascript file. For example

<script type="text/javascript" src="scripts/ctl_owrite.js?version=20181204”></script>

If this parameter is changed every time the script is updated, it achieves two things. Firstly, it modifies the HTML file and secondly (more importantly) it changes the ‘url’ of the javascript file so the browsers can no longer use the cached javascript files. We have tested this solution with a number of mainstream browsers. Using this trick could potentially prevent a support nightmare.

Important Links

News: https://home.brainydata.com/news.htm

Products & pricing: https://products.brainydata.com

Demo/Examples Downloads: https://demos.brainydata.com/download.htm

Github: https://github.com/BrainyData

Sponsors: https://home.brainydata.com/sponsors.htm

Feedback: https://home.brainydata.com/customers.htm

Online Documentation: https://supportpublic.brainydata.com/documentation.htm

Technical notes: https://supportpublic.brainydata.com/technotes.htm

Support Request Form: visit https://supportpublic.brainydata.com and click “Software Downloads”

Software Downloads: visit https://supportpublic.brainydata.com and click “Contact Support”