User Interface - Bug #2929

Feature # 1811 (Closed): implement the AJAX client driver

test chrome browser support

12/12/2015 07:27 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Sergey Ivanovskiy	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Bug #2964: fix key processing for web chui in Fir...	**New**	**01/21/2016**

History

#1 - 12/12/2015 07:28 AM - Greg Shah

Please test the web client from the Chrome browser. Report any found issues here. Resolve those issues. Anything risky can be posted here as a diff. If the changes are not risky, they can be directly checked into 1811t.

For testing, please run through the standalone GUI testcases list (the same ones we use for regression testing GUI). All testing should be done on branch 1811t.

#2 - 12/15/2015 07:43 AM - Sergey Ivanovskiy

The test with Google Chrome Version 47.0.2526.80 (64-bit) is failed throwing this root cause exception

```
p2j.js:138 Uncaught TypeError: Cannot set property style of #<HTMLElement> which has only a getter
setOwnPropertiesTo @ p2j.js:138
me.createCanvas @ p2j.js:168
Window @ p2j.screen.js:353
createWindowInternal @ p2j.screen.js:1955
p2j.screen.me.createWindow @ p2j.screen.js:1927
ws.onmessage @ p2j.socket.js:461
```

For Chrome the "style" property is read only.

#3 - 12/15/2015 02:56 PM - Sergey Ivanovskiy

- File chrome_style_issue_1.txt added

Committed revision 10963 fixed the "style of HTML Element has only a getter" issue for Chrome browser.

#4 - 12/15/2015 02:58 PM - Sergey Ivanovskiy

- File chrome_image_issue.png added

Have started to work on this issue: for Chrome the image drawing works incorrectly. Found that getImageData(x, y, w, h) works differently for Chrome in the case of x and y are not integer values.

#5 - 12/15/2015 05:38 PM - Sergey Ivanovskiy

- File chrome_images_issue_1.txt added

Committed revision 10964 fixes the getImageData implementation for Chrome. The final result is not with a precise accuracy but it is suitable for occasional presentations.

#6 - 12/16/2015 08:43 AM - Sergey Ivanovskiy

Committed revision 10965 adds a Chrome's fix for applyCompositeMode.

#7 - 12/17/2015 10:58 AM - Greg Shah

Code Review Task Branch 1811t Revision 10963

1. There is this code in toProperties():

```
            if (typeof styleObj[prop] !== "object")
            {
               clsDef += (prop + ":" + styleObj[prop] + ";");
            }
```

In JS, typeof null === "object". How does this code react when styleObj[prop] is null?

2. To be honest, I'm a bit surprised that Chrome acts this way. I doubt that Chrome has all style properties as readonly. That is actually the recommended way to modify properties (much preferred over setAttribute()). Perhaps just this particular property is read-only?

I wonder if this is a place where DOJO may have some helpful features? Consider this:

https://dojotoolkit.org/reference-guide/1.7/dojo/style.html

#8 - 12/17/2015 11:03 AM - Greg Shah

Code Review Task Branch 1811t Revision 10964

I don't understand the change to CanvasRenderer.prototype.drawImage(). The comments say that "Changed to work around that the Chrome's implementation of getImageData rounds function parameters.". But the change implemented means that our JS code is rounding these values. How

is this "working around" the rounding that Chrome is doing?

> The final result is not with a precise accuracy but it is suitable for occasional presentations.

This doesn't seem right. Please explain why this is safe.

#9 - 12/17/2015 11:04 AM - Greg Shah

Code Review Task Branch 1811t Revision 10965

I have the same concerns for the applyCompositeMode() changes as I have for the drawImage() changes. These changes don't seem safe or correct.

#10 - 12/17/2015 03:20 PM - Sergey Ivanovskiy

> Code Review Task Branch 1811t Revision 10963
>
> 1. There is this code in toProperties():
>
> [...]
>
> In JS, typeof null === "object". How does this code react when styleObj[prop] is null?

All its usages are safe now and it is supposed that styleObj is a plain object with simple key-value pairs. Planning to fix it. This function is enough for all our style usages.

> 2. To be honest, I'm a bit surprised that Chrome acts this way. I doubt that Chrome has all style properties as readonly. That is actually the recommended way to modify properties (much preferred over setAttribute()). Perhaps just this particular property is read-only?

Yes, the Chrome and Opera have not implemented yet CSSStyleDeclaration from CSSOM (http://www.w3.org/TR/cssom/) but Firefox has.

> I wonder if this is a place where DOJO may have some helpful features? Consider this:
>
> https://dojotoolkit.org/reference-guide/1.7/dojo/style.html

Agree with you, but would like to remember that you are planning to eliminate dojo in our p2j project.

#11 - 12/17/2015 03:35 PM - Sergey Ivanovskiy

Code Review Task Branch 1811t Revision 10964

I don't understand the change to CanvasRenderer.prototype.drawImage(). The comments say that "Changed to work around that the Chrome's implementation of getImageData rounds function parameters.". But the change implemented means that our JS code is rounding these values. How is this "working around" the rounding that Chrome is doing?

> The final result is not with a precise accuracy but it is suitable for occasional presentations.

This doesn't seem right. Please explain why this is safe.

I think that it is a bug or an intended behavior of Chrome's implementation of getImageData. If getImageData gets float coordinates it probably moves the target image area systematically right to the nearest integer or doesn't work with floats at all. I mean 0.5 rounds as 1. I need a separate test in order to prove that Chrome don't work with float coordinates.
This fix ctx.getImageData(Math.round(xo), Math.round(yo), width, height); is important for images drawing.

#12 - 12/21/2015 10:03 AM - Sergey Ivanovskiy

Greg, please look at this test.
Found that this simple test ctx.getImageData(x, y, 100, 100); gives different results for Firefox and Google Chrome iff x and y are float numbers.

```
var img0 = ctx.getImageData(0.5, 0.5, 100, 100);
console.debug(img0);
```

The output for Firefox gives ImageData { width: 100, height: 100, data: Uint8ClampedArray[40000] }, but the output for Google Chrome ImageData { width: 101, height: 101, data: Uint8ClampedArray[40802] }. Thus the actual image width and height are changed for the Google Chrome if x and y are float numbers. Because of this strange behavior images are corrupted for the Google Chrome. It explains why this fix ctx.getImageData(Math.round(xo), Math.round(yo), width, height); works for Google Chrome.

#13 - 12/22/2015 01:55 PM - Sergey Ivanovskiy

- File chrome_style_issue_2.txt added

Greg, please review the changes revision 10968.

#14 - 12/30/2015 10:20 PM - Greg Shah

> Agree with you, but would like to remember that you are planning to eliminate dojo in our p2j project.

We won't eliminate DOJO if it has real value.

#15 - 12/30/2015 10:24 PM - Greg Shah

> The output for Firefox gives ImageData { width: 100, height: 100, data: Uint8ClampedArray[40000] }, but the output for Google Chrome ImageData { width: 101, height: 101, data: Uint8ClampedArray[40802] }. Thus the actual image width and height are changed for the Google Chrome if x and y are float numbers.

OK, I understand the need for the change when running in Chrome.

> The final result is not with a precise accuracy but it is suitable for occasional presentations.

I'm still concerned about this comment. In which cases will our output be imprecise?

#16 - 12/30/2015 10:25 PM - Greg Shah

Code Review Task Branch 1811t Revision 10968

I'm OK with the changes (they do make things better).

However, I still would like you to investigate if the DOJO class add/remove utility functions would be an improvement.

#17 - 01/02/2016 04:28 PM - Sergey Ivanovskiy

- File demo_widgets_chrome.png added

Greg Shah wrote:

> The output for Firefox gives ImageData { width: 100, height: 100, data: Uint8ClampedArray[40000] }, but the output for Google Chrome ImageData { width: 101, height: 101, data: Uint8ClampedArray[40802] }. Thus the actual image width and height are changed for the Google Chrome if x and y are float numbers.

OK, I understand the need for the change when running in Chrome.

> The final result is not with a precise accuracy but it is suitable for occasional presentations.

I'm still concerned about this comment. In which cases will our output be imprecise?

For Google Chrome ./demo/demo_widgets.p looks like this rough picture. If my observations are correct than the images on the caption buttons are displayed one pixel closer to the right sides, but the image on the combo box button is displayed one pixel closer to the left side.

#18 - 01/03/2016 12:10 PM - Sergey Ivanovskiy

Greg Shah wrote:

> Code Review Task Branch 1811t Revision 10968
>
> I'm OK with the changes (they do make things better).
>
> However, I still would like you to investigate if the DOJO class add/remove utility functions would be an improvement.

Agree with you. Have checked that already loaded DOJO base module can help to substitute function setStyleForElement(element, styleObj) with this one

```
function setStyleForElement(element, styleObj)
{
   dojo.style(element, styleObj);
}
```

and to get rid of function toProperties(styleObj).

#19 - 01/04/2016 10:09 AM - Sergey Ivanovskiy

Committed revision 10971 reuses DOJO style API. Committed revision 10970 fixes the cursor positioning within the web editor.

#20 - 01/05/2016 11:37 AM - Greg Shah

Code Review Task Branch 1811t Revision 10971

The change is good.

#21 - 01/05/2016 11:40 AM - Greg Shah

> For Google Chrome ./demo/demo_widgets.p looks like this rough picture. If my observations are correct than the images on the caption buttons are displayed one pixel closer to the right sides, but the image on the combo box button is displayed one pixel closer to the left side.

If I understand correctly, the Chrome support today is still flawed (the rounding implementation is not a 100% solution). Please look for a resolution to this issue.

#22 - 01/11/2016 12:41 PM - Sergey Ivanovskiy

- File chrome_copy_image_1.txt added

Greg, please review the committed revision 10978, it looks like it fixes the Chrome's images issue at all.

#23 - 01/11/2016 03:38 PM - Greg Shah

Code Review Task Branch 1811t Revision 10978

The changes are fine.

#24 - 01/11/2016 03:39 PM - Greg Shah

Is there any other testing needed for this task?

Are there any other issues to fix?

#25 - 01/12/2016 12:00 PM - Sergey Ivanovskiy

Greg Shah wrote:

> Is there any other testing needed for this task?
>
> Are there any other issues to fix?

The images are fixed but there exists new bug with the incorrect title font. The window title for Chrome is displayed in the normal font style and if the window is resized, then it is changed to the bold style and continues to be the bold. Thus the embedded Tahoma font is not applied to the title drawing at the moment.

#26 - 01/12/2016 04:00 PM - Sergey Ivanovskiy

Committed revision 10979 in order to check if the given embedded font has been already defined. On the rev 10979 I can't reproduce the title bug for Google Chrome and for Chromium, thus now #2929 has no other known issues to fix.

#27 - 01/13/2016 06:02 AM - Sergey Ivanovskiy

- File titleInChrome.png added

Sergey Ivanovskiy wrote:

> Committed revision 10979 in order to check if the given embedded font has been already defined. On the rev 10979 I can't reproduce the title bug for Google Chrome and for Chromium, thus now #2929 has no other known issues to fix.

The correct statement is that the title bug for Google Chrome and Chromium can be still reproduced on the clean browser tab page only once and the next login on the same page gives the correct title font. This bug is specific for Google Chrome and Chromium.

#28 - 01/13/2016 07:14 AM - Sergey Ivanovskiy

The following article and its linked resources can be useful to the title font issue https://dev.opera.com/articles/better-font-face/

#29 - 01/13/2016 08:18 AM - Sergey Ivanovskiy

The committed revision 10981 forces the embedded font loading.

#30 - 01/13/2016 09:14 AM - Greg Shah

Code Review Task Branch 1811t Revision 10981

The changes are good.

Can I close this task?

#31 - 01/13/2016 09:23 AM - Sergey Ivanovskiy

There are no known bugs.

#32 - 01/13/2016 09:43 AM - Greg Shah

- Status changed from New to WIP

Did you test clipboard operations?

You've tested both the 454 testcase as well as some standalone testcases?

#33 - 01/13/2016 10:26 AM - Sergey Ivanovskiy

I haven't tested yet with all standalone tests that are usually used, only with demo_widgets.p. The clipboard operations work but my test program needs to be repaired because

```
keystrokes.js:191 Uncaught NotSupportedError: Failed to execute 'createEvent' on 'Document': The provided even
t type ('KeyEvents') is invalid.
```

Thus I can't check the keys test. I am going to repair this test if you have no objections. Your question gets the target. New bug is BACKSPACE doesn't work for Chrome in fill-in and editor.

#34 - 01/13/2016 10:30 AM - Greg Shah

> I am going to repair this test if you have no objections.

Certainly, please do.

> New bug is BACKSPACE doesn't work for Chrome in fill-in and editor.

OK. Please fix it. Now is the time to find and fix these browser differences.

#35 - 01/13/2016 02:41 PM - Sergey Ivanovskiy

Fixed the backspace delivery committed rev 10983 and fixed web keys simulation for Chrome and IE - testcase project committed rev. 1452.

#36 - 01/14/2016 10:11 AM - Greg Shah

Code Review Task Branch 1811t Revision 10983

The changes are fine.

Let me know when you believe your testing is complete.

#37 - 01/14/2016 04:28 PM - Sergey Ivanovskiy

> Code Review Task Branch 1811t Revision 10983
>
> The changes are fine.
>
> Let me know when you believe your testing is complete.

Ok, found that for Chrome the manual tests are different from the automatic tests. The automatic tests are passed for CTRL + SHIFT + key but manual tests are failed. Also manual tests for Alt + Key are failed for Chrome. It happens because keypress for Chrome is not generated for CTRL + SHIFT + key or Alt + key, but the automatic tests simulate keypress events.

#38 - 01/14/2016 04:45 PM - Sergey Ivanovskiy

The same problem is in IE11, thus it needs to fix the keys processing for Chrome and IE. The automatic tests are repaired but the events processing of these browsers are different from the Firefox, thus these automatic tests work only for the Firefox. IE doesn't generate keyup events for Alt + key combinations, thus the logic becomes more complicated.

#39 - 01/15/2016 07:24 AM - Greg Shah

> thus the logic becomes more complicated.

Understood.

#40 - 01/18/2016 09:29 AM - Sergey Ivanovskiy

There are exception cases in the case the Chrome handles the Progress special characters @, ^, _ , \,].
The usual generated events on CTRL + special character key press are these ones

```
keydown  keyCode=17        which=17        charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=1 repeat=false
keydown  keyCode=50  (2)   which=50  (2)   charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keyup    keyCode=50  (2)   which=50  (2)   charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keyup    keyCode=17        which=17        charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=1 repeat=false
```

But ']' and '\' are exceptions and the Chrome produces these events:

```
keydown  keyCode=17        which=17        charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=1 repeat=false
keydown  keyCode=221       which=221       charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keypress keyCode=29        which=29        charCode=29
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keyup    keyCode=221       which=221       charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keyup    keyCode=17        which=17        charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=1 repeat=false
```

and

```
keydown  keyCode=17        which=17        charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=1 repeat=false
keydown  keyCode=220       which=220       charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keypress keyCode=28        which=28        charCode=28
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keyup    keyCode=220       which=220       charCode=0
         shiftKey=false ctrlKey=true  altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=false
keyup    keyCode=17        which=17        charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=1 repeat=false
```

The other issue is how to prevent the default actions that can be associated with keys combinations, for example CTRL + 'S' (Save As) or ALT + 'S'(History).
For new version of Firefox 43.0.4 Ubuntu 15.10 the combination ALT + 'S' can't be prevented, but CTRL + 'S' can be stopped. It looks like the browser's mess.

#41 - 01/18/2016 11:18 PM - Sergey Ivanovskiy

There is the problem with new Firefox, because it is impossible to prevent the default browser action on the pressed Alt + key. Thus if the application has the same shortcuts, on the forum it is advised to change the keys combination to use SHIFT + Alt + key instead of Alt + key and it is considered as a good solution.

#42 - 01/19/2016 07:53 AM - Greg Shah

I understand about the restriction for certain accelerator combinations.

Please add a note to #2675 which lists the combinations which are excluded. Please make a list for each browser.

#43 - 01/19/2016 08:03 PM - Sergey Ivanovskiy

Committed rev 10985 (1811t) and rev 1455 (testcases) fixes the keyboard events processing for Chrome and IE, but Web Chui keyboard processing is not ready. Can we rely on linux-chui.txt from testcases/uast/keyboards in order to check the key processing in the character mode?

#44 - 01/21/2016 04:13 AM - Sergey Ivanovskiy

Sergey Ivanovskiy wrote:

> Committed rev 10985 (1811t) and rev 1455 (testcases) fixes the keyboard events processing for Chrome and IE, but Web Chui keyboard processing is not ready. Can we rely on linux-chui.txt from testcases/uast/keyboards in order to check the key processing in the character mode?

Greg, 10985 (10986 cleanup) did some reasonable changes in module p2j.keymap.js to fix Progress 4GL codes generated by function mapKey(evt) for letters and Web Chui key processing was involved.
Currently the Web Chui keyboard module hasn't generated all 4GL keys codes correctly. Do we require to support all modern Firefox, Chrome, IE and Safari for Web Chui? The question is due to the logic implemented doesn't generate all codes correctly. Does it need to run MAJIC application to test Web Chui? I think that MAJIC uses only some subset of all possible keys and current logic covers the required keys.

#45 - 01/21/2016 10:29 AM - Greg Shah

> Can we rely on linux-chui.txt from testcases/uast/keyboards in order to check the key processing in the character mode?

Yes, I think so.

> Do we require to support all modern Firefox, Chrome, IE and Safari for Web Chui? The question is due to the logic implemented doesn't generate all codes correctly.

If MAJIC is working properly, then I think we can defer this until later. Please do some testing in MAJIC to ensure that it is working.

Also: create a new task (link it as a related task to this one and also to #1811). The new task should be named "fix key processing for web chui in Firefox, Chrome, IE and Safari". Please document the known issues and the expected work needed to resolve the task.

#46 - 01/21/2016 10:47 PM - Sergey Ivanovskiy

- File client_sbi_23757.log added

- File server.log added

Greg, I installed MAJIC accessing to the local Postgres database. It seems that the MAJIC server is running with the properly installed database gso_20090909_staging2. But for the current branch 1811t the function browser doesn't work for the chui client and also for the web client. For the chui client if the [?] key is pressed, then the screen is splashed for a moment. There are no exceptions on the server logs and the client logs. It seems that the others functions and keys work properly. Are there the [?] key processing failures or the function browser failures?

#47 - 01/22/2016 07:55 AM - Greg Shah

The only bug I know about is #2733. That just is a drawing problem, it doesn't stop the function browser from working. What you are seeing is probably a regression. You can try the same thing in the trunk to confirm.

#48 - 01/22/2016 11:47 AM - Sergey Ivanovskiy

> The only bug I know about is #2733. That just is a drawing problem, it doesn't stop the function browser from working. What you are seeing is probably a regression. You can try the same thing in the trunk to confirm.

Testing with trunc using my local environment and using the devsrv01 environment confirms the issue that the function browser isn't displayed on pressing the [?] key.

#49 - 01/25/2016 03:17 AM - Sergey Ivanovskiy

> Let me know when you believe your testing is complete.

I think that Chrome differences is fixed in general with possible bugs in keys processing due to some keys, for example, '? and /' has 63 and 47 ASCII codes but ? appears only if we press the shift modifier, 'Caps Lock' for Firefox and Chrome can't give a knowledge about the keys state. Thus the key sequence Ctrl + Shift + '? and /' is really the same as
Ctrl + Shift + '/', but it is not possible to press something like this Ctrl + Shift + '?', because the browser's scancode is 191 for the both cases.

#50 - 01/25/2016 08:39 AM - Greg Shah

You are talking about the keydown events here? I assume the keypress event generates the correct character, but this is too late for us?

http://unixpapa.com/js/key.html

#51 - 01/25/2016 11:33 AM - Sergey Ivanovskiy

> You are talking about the keydown events here? I assume the keypress event generates the correct character, but this is too late for us?
>
> http://unixpapa.com/js/key.html

One of the differences between browsers is that the Chrome and IE don't generate "keypress" events if ALT, CTRL or SHIFT modifier is set, but the Firefox generates them. The second is the default actions. If we invoke evt.preventDefault(); on "keydown" events, then some of the default actions can be prevented. Please see #2675-2. But there is the following side effect, the "keypress" events don't occur. This case is handled by our code, but there are cases that can't be resolved without "keypress" events. For example, "Caps Lock" is on or off. Nevertheless the Chrome doesn't generate "keypress" events if one of the modifiers is pressed. I will try to resolve these issues.

#52 - 01/25/2016 03:57 PM - Sergey Ivanovskiy

Committed revision 10987 fixes CTRL + CTRL + key with the exception set. For example, if we press left and right control and 'Z', then 'Z' is ignored by Firefox and Chrome.
Propose to close the tasks 2929 and 2930 and to support only the required keys combinations for the converted application logic. In general all keys are supported.

#53 - 01/25/2016 04:04 PM - Greg Shah

- Status changed from WIP to Closed

- % Done changed from 0 to 100

#54 - 01/25/2016 04:06 PM - Greg Shah

Please add any additional found key exclusions (e.g. CTRL + CTRL + Z) to #2675.

#55 - 11/16/2016 12:12 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App

Files

chrome_style_issue_1.txt	8.73 KB	12/15/2015	Sergey Ivanovskiy
chrome_image_issue.png	30.1 KB	12/15/2015	Sergey Ivanovskiy
chrome_images_issue_1.txt	2.29 KB	12/15/2015	Sergey Ivanovskiy
chrome_style_issue_2.txt	956 Bytes	12/22/2015	Sergey Ivanovskiy
demo_widgets_chrome.png	18.6 KB	01/02/2016	Sergey Ivanovskiy
chrome_copy_image_1.txt	5.03 KB	01/11/2016	Sergey Ivanovskiy
titleInChrome.png	359 KB	01/13/2016	Sergey Ivanovskiy
server.log	24.6 KB	01/22/2016	Sergey Ivanovskiy
client_sbi_23757.log	41.3 KB	01/22/2016	Sergey Ivanovskiy