

## User Interface - Bug #2931

### test Safari browser support

12/12/2015 07:30 AM - Greg Shah

<b>Status:</b>	Closed	<b>Start date:</b>	
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>	Sergey Ivanovskiy	<b>% Done:</b>	100%
<b>Category:</b>		<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>	Cleanup and Stabilization for GUI	<b>case_num:</b>	
<b>billable:</b>	No		
<b>vendor_id:</b>	GCD		
<b>Description</b>			
<b>Related issues:</b>			
Related to User Interface - Feature #1811: implement the AJAX client driver			
		<b>Closed</b>	<b>11/01/2013 03/06/2014</b>

### History

#### #1 - 12/12/2015 07:30 AM - Greg Shah

Please test the web client from the Safari 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 - 03/31/2016 12:46 PM - Greg Shah

- Target version changed from Milestone 12 to Milestone 16

- Parent task deleted (#1811)

#### #3 - 05/19/2016 10:04 AM - Sergey Ivanovskiy

Committed revision 11044(3084a) substitutes "const" statements to "var"s. Encountered new issue that self-signed certificates aren't supported by Web Socket Client API implemented by Safari (OS X El Capitan 10.11 on the Mac mini)

```
[Error] WebSocket network error: OSStatus Error -9807: Invalid certificate chain
[Error] Connection closed with the returned code 1006
      error (p2j.screen.js:2943)
      onclose (p2j.socket.js:2561)
```

Working on this issue. The detailed system logs

```
19/05/16 19:39:29,940 com.apple.WebKit.WebContent[965]: CFNetwork SSLHandshake failed (-9807)
19/05/16 19:39:29,995 com.apple.WebKit.WebContent[965]: NSURLSession/NSURLConnection HTTP load failed (kCFStreamErrorDomainSSL, -9813)
19/05/16 19:39:29,997 com.apple.WebKit.WebContent[965]: [19:39:29.997] HTTPRequest figHttpRequestDidFailCallback: Network error: domain=kCFErrorDomainCFNetwork, code=-1202, Error Domain=kCFErrorDomainCFNetwork Code=-1202
"The certificate for this server is invalid. You might be connecting to a server that is pretending to be 192.168.1.35 which could put your confidential information at risk." UserInfo={NSErrorFailingURLStringKey=https://192.168.1.35:7449/common/beep.wav, NSLocalizedRecoverySuggestion=Would you like to connect to the server anyway?, _kCFNetworkCFStreamSSLErrorOriginalValue=-9813, kCFStreamPropertySSLPeerCertificates=(
    "<SecCertificate 0x7fa4cleae380 [0x7fff77d64440]>",
    "<SecCertificate 0x7fa4cleab510 [0x7fff77d64440]>"
), _kCFStreamPropertySSLClientCertificateState=0, kCFStreamPropertySSLPeerTrust=<SecTrust 0x7fa4cleb2a30 [0x7fff77d64440]>, NSLocalizedDescription=The certificate for this server is invalid. You might be connecting to a server that is pretending to be 192.168.1.35 which could put your confidential information at risk., _kCFStr
```

```
eamErrorDomainKey=3, NSErrorFailingURLKey=https://192.168.1.35:7449/common/beep.wav, _kCFStreamErrorCodeKey=-9813} for https://192.168.1.35:7449/common/beep.wav
19/05/16 19:39:30,000 com.apple.WebKit.WebContent[965]: [19:39:29.999] FigDCP_GetCacheFileVersion signalled error=-12540 (kFigDiskCacheProviderError_InvalidCacheFile) (unsupported file version) at /Library/Caches/com.apple.xbs/Sources/CoreMedia/CoreMedia-1731.15.204/Prototypes/FigByteStreamPrototypes/FigLimitedDiskCacheProvider.c
line 384
```

#### #4 - 05/19/2016 02:59 PM - Sergey Ivanovskiy

The P2J standard server certificate must be set as a trusted one at the first access to the login page. If it isn't done, then it needs to remove its identity from the Safari cache with help of Keychain Access.app. If the identity is removed, then accessing to the login page invokes the Safari to open the certificate warning dialog. I will provide images with instructions to follow later.

#### #5 - 05/19/2016 03:02 PM - Sergey Ivanovskiy

Committed revision 11045 fixed drawings of images. It seems that Chrome and Safari browsers have the same drawing issues due to they are WebKit-based browsers. Now I am using <https://www.nomachine.com/> to manage the parkland of devices. It can be installed on each computer in the local network (for Windows, Mac Os X, Linux, ...) and can be managed from the box without complicated settings.

#### #6 - 05/30/2016 06:07 PM - Sergey Ivanovskiy

For Safari CTRL + LETTER keystrokes raises keypress event with charCode that is calculated by  $\text{ctrlCode} = \text{keyDownCode} \& 0\text{xffbf}$ . Thus pressing CTRL+C invokes keypress with  $\text{charCode} = \text{keyCode} = \text{which} = 3$  and keypress can't be suppressed for CTRL+C, CTRL+V, CTRL+X, but it can be suppressed for CTRL+B as an example. The copy, cut and paste actions are invoked by pressing COMMAND+C, COMMAND+X and COMMAND+V respectively, where the COMMAND key corresponds to the left window or to the right window key on Windows keyboards. The COMMAND key is detected by the metaKey event flag.  $\text{keyDownCode}('C') \& 0\text{xffbf} = 67 \& 0\text{xffbf} = 0000\ 0000\ 0100\ 0011 \& 0\text{xffbf} = 0000\ 0000\ 0000\ 0001$ . Google Chrome and IE11 don't raise a keypress event for CTRL+C and Firefox yields keypress with  $\text{charCode} = 99$ , otherwise Safari produces a keypress event with  $\text{charCode} = \text{keyCode} = \text{which} = 3$ .

#### #7 - 05/31/2016 04:20 AM - Sergey Ivanovskiy

- File *testKey.html* added

Note 6 states incorrectly that keypress event can't be suppressed for CTRL+C, CTRL+V, CTRL+X with Safari. I am using simple test page developed by <http://unixpapa.com>. Since CTRL+C, CTRL+V, CTRL+X produce the target copy, paste and cut actions, pressing these keys aren't suppressed for IE11, Chrome and Firefox. It confuses me during testing.

#### #8 - 05/31/2016 05:43 AM - Sergey Ivanovskiy

I didn't find special Progress 4GL rules to map COMMAND + key, where COMMAND is a LEFT\_WINDOW key or a RIGHT\_WINDOW key. These keys have the same code as '[' and ']' keys respectively. Thus we can map COMMAND+C, COMMAND+X and COMMAND+V to CTRL+C, CTRL+X and CTRL+V Progress 4GL keys codes, otherwise produce separate 4GL codes for pressing a COMMAND key and an any other key except 'C', 'X' and 'V'.  
Constantin, what do you think it is a correct solution?

#### #9 - 05/31/2016 05:55 AM - Sergey Ivanovskiy

The other question is that should we map the others useful keyboard shortcuts to 4GL codes with the same meaning to be friendly with Mac OS users.

Please look at these shortcuts:

<https://support.apple.com/en-us/HT201236>

For an example, should we map COMMAND + LEFT\_ARROW to HOME 4GL Progress code?

#### #10 - 05/31/2016 06:18 AM - Sergey Ivanovskiy

- File *demo\_widgets\_mac\_os\_x.png* added

My first expression of using MAC OS X is very positive and I like the hardware implementation of Mac mini too. Please look at the *demo\_widgets.p* on Mac OS X with Safari web client.

#### #11 - 05/31/2016 06:33 AM - Sergey Ivanovskiy

Greg, what do you think it is necessarily to map MAC OS shortcuts with new modifier key COMMAND to 4GL codes with the same meaning? For an example, to go to the end of a line we need to press COMMAND + RIGHT\_ARROW, but for Windows and Linux we press END key.

#### #12 - 05/31/2016 06:53 AM - Sergey Ivanovskiy

The most different behavior of Safari can be observed if we press and hold some key. The Safari produces these codes if keydown events are suppressed

```
keydown  keyCode=83 (S)  which=83 (S)  charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229    which=229    charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229    which=229    charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229    which=229    charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229    which=229    charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229    which=229    charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229    which=229    charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keyup    keyCode=83 (S)  which=83 (S)  charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
textInput data=ß
ß
```

with opened window char select dialog that is displayed on keyup event. The repeat flag isn't set and keyCode=229 is the common code for all pressed keys.

If keydown events are not suppressed, then it yields these logs

```
keydown  keyCode=65 (A)  which=65 (A)  charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keypress keyCode=97 (a)  which=97 (a)  charCode=97 (a)
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
textInput data=a
akeydown keyCode=229    which=229    charCode=0
```

```

      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229      which=229      charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229      which=229      charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=229      which=229      charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keyup    keyCode=65 (A)    which=65 (A)    charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
textInput data=a
a

```

The difference is only textInput and keypress are raised one time.

#### #13 - 05/31/2016 06:59 AM - Sergey Ivanovskiy

The repeat keys behavior of Safari can be observed if we access a physical keyboard of the target device. In the case of NoMachine GUI client pressing and holding one key produces fake keys up events.

#### #14 - 05/31/2016 07:09 AM - Sergey Ivanovskiy

The described in notes 12 and 13 Safari behavior is related to the different input sources. For my environment English and Russian are installed.

#### #15 - 05/31/2016 07:25 AM - Sergey Ivanovskiy

It looks like a system bug because switching input sources and removing additional ones can produce correct browser events without the select char dialog if keydown events are suppressed

```

keydown  keyCode=67 (C)    which=67 (C)    charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keyup    keyCode=67 (C)    which=67 (C)    charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=67 (C)    which=67 (C)    charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keyup    keyCode=67 (C)    which=67 (C)    charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keydown  keyCode=67 (C)    which=67 (C)    charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined
keyup    keyCode=67 (C)    which=67 (C)    charCode=0
      shiftKey=false ctrlKey=false altKey=false metaKey=false
      key=undefined char=undefined location=0 repeat=undefined

```

But if we don't suppress keydown events then incorrect sequence can be observed with select char dialog on keyup

```
keydown  keyCode=67 (C)  which=67 (C)  charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keypress keyCode=99 (c)  which=99 (c)  charCode=99 (c)
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
textInput data=c
ckeydown  keyCode=229      which=229      charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
keyup     keyCode=67 (C)  which=67 (C)  charCode=0
         shiftKey=false ctrlKey=false altKey=false metaKey=false
         key=undefined char=undefined location=0 repeat=undefined
textInput data=ç
ç
```

#### #16 - 05/31/2016 07:30 AM - Sergey Ivanovskiy

There are graphical keyboards settings: Key Repeat and Delay Until Repeat, but the keyboard type is selected with the help of the wizard by listening the user's input on the target requested keys. At the moment I don't know how to deal with char select dialog, but the other differences can be overcome.

#### #17 - 05/31/2016 09:05 AM - Sergey Ivanovskiy

Planning to rebase 3084a.

#### #18 - 05/31/2016 09:35 AM - Greg Shah

I didn't find special Progress 4GL rules to map COMMAND + key, where COMMAND is a LEFT\_WINDOW key or a RIGHT\_WINDOW key. These keys have the same code as 'I' and 'J' keys respectively. Thus we can map COMMAND+C, COMMAND+X and COMMAND+V to CTRL+C, CTRL+X and CTRL+V Progress 4GL keys codes, otherwise produce separate 4GL codes for pressing a COMMAND key and an any other key except 'C', 'X' and 'V'.

The 4GL has never been available on MacOS. Thus, it doesn't have any matching values that we can test and implement. For this reason, we want to make a direct and simple (at least as simple as possible) mapping between what is possible on the Mac keyboard to what is needed in the 4GL.

For example, can we map COMMAND to the CTRL key (for everything, not just for cut/copy/paste)? If this is possible, then it can be an easy thing for a user to understand and use.

**#19 - 05/31/2016 10:00 AM - Sergey Ivanovskiy**

Yes, planning to map COMMAND to CTRL by processing COMMAND + keystrokes as it is CTRL+keystrokes.  
3084a is rebased to revision 11061 from trunc 11040.

**#20 - 05/31/2016 10:07 AM - Greg Shah**

Yes, planning to map COMMAND to CTRL by processing COMMAND + keystrokes as it is CTRL+keystrokes.

OK, good. After doing this, we must not re-use COMMAND as some other replacement (e.g. COMMAND + RIGHT\_ARROW = END) since CTRL + RIGHT\_ARROW will have an expected set of key events that may be generated.

What is the list of keys on the standard PC keyboard that cannot be represented using the Mac keyboard?

**#21 - 05/31/2016 10:22 AM - Sergey Ivanovskiy**

Yes, planning to map COMMAND to CTRL by processing COMMAND + keystrokes as it is CTRL+keystrokes.

OK, good. After doing this, we must not re-use COMMAND as some other replacement (e.g. COMMAND + RIGHT\_ARROW = END) since CTRL + RIGHT\_ARROW will have an expected set of key events that may be generated.

What is the list of keys on the standard PC keyboard that cannot be represented using the Mac keyboard?

Do you mean that if we replace COMMAND by CTRL, then COMMAND+RIGHT\_ARROW action becomes CTRL+RIGHT\_ARROW and can't represent its MAC OS X action to move the insertion point to the end of the current line. There are many such keys if we follow this document <https://support.apple.com/en-us/HT201236>, but there are common actions such that CTRL+F, CTRL+C, CTRL+X, CTRL+V.

**#22 - 05/31/2016 11:08 AM - Greg Shah**

Do you mean that if we replace COMMAND by CTRL, then COMMAND+RIGHT\_ARROW action becomes CTRL+RIGHT\_ARROW and can't represent its MAC OS X action to move the insertion point to the end of the current line.

Yes, I mean exactly this.

The 4GL has some important CTRL + combinations that must be supported. Individual apps can and do assign their own key events that depend on the CTRL key. We can't arbitrarily map COMMAND + to other Mac-specific events because this can "hide" 4GL or application events.

There are many such keys if we follow this document <https://support.apple.com/en-us/HT201236>, but there are common actions such that CTRL+F, CTRL+C, CTRL+X, CTRL+V.

We don't need to support the common Mac key combinations. I am only worried about supporting the 4GL processing that is needed.

I guess OPTION needs to be used for ALT?

How does one do F1 - F12? How to do HOME, END, INSERT, DELETE, BACKSPACE, PAGE-UP, PAGE-DOWN...? These are things that the 4GL will depend on, since the 4GL was designed for PC keyboards. Don't worry about doing things "the Mac way". We only need to worry about how a user can properly use the converted 4GL app.

**#23 - 06/02/2016 03:42 PM - Sergey Ivanovskiy**

I guess OPTION needs to be used for ALT?

Yes.

How does one do F1 - F12? How to do HOME, END, INSERT, DELETE, BACKSPACE, PAGE-UP, PAGE-DOWN...? These are things that the 4GL will depend on, since the 4GL was designed for PC keyboards. Don't worry about doing things "the Mac way". We only need to worry about how a user can properly use the converted 4GL app.

Safari generates the same codes as Firefox for keys from F1 to F12, the default actions on these keys can be prevented on keydown events. Pressing these keys ALT, HOME, END, DELETE, BACKSPACE, PAGE-UP, PAGE-DOWN produce keydown and keyup events, but these keys BREAK, Scroll Lock and Print Screen keys don't generate any event for Safari and can't be detected. Pressing Insert generates only keyup event. LEFT, UP, RIGHT, DOWN arrows, ESC, ENTER can be detected. Caps Lock throws only one event depending on its state. In my environment I encountered the following autorepeat problem that sometimes pressing and holding the typed key produces incorrect sequence (note 15). It looks like a keyboard driver bug or incorrect software configuration. I think the required keys and their combinations can be correctly detected.

**#24 - 06/03/2016 05:14 AM - Sergey Ivanovskiy**

These commands `document.execCommand("copy|cut|paste")` are not supported by Safari, thus we must prompt a user to use keyboard shortcuts: `COMMAND+C`, `COMMAND+X`, `COMMAND+V`.

**#25 - 06/03/2016 06:23 AM - Sergey Ivanovskiy**

- *File 3080\_clipboard\_10.txt added*

Please review the committed revision 11064. There still exists the autorepeat issue even the automatic switch to a document's input source is turned off and correct spelling automatically is off too and there is the unique input source pointed to US language.

**#26 - 06/03/2016 07:09 AM - Sergey Ivanovskiy**

It seems that Safari sends `keyDown` events with 229 key code for autorepeats.

<http://stackoverflow.com/questions/25043934/is-it-ok-to-ignore-keydown-events-with-keycode-229>

**#27 - 06/03/2016 11:43 AM - Sergey Ivanovskiy**

It seems that this document demystifies the root of the 229 code

<https://lists.w3.org/Archives/Public/www-dom/2010JulSep/att-0182/keyCode-spec.html>

**#28 - 06/03/2016 02:15 PM - Sergey Ivanovskiy**

- *File 3080\_clipboard\_10.txt added*

Greg, please review committed revision 11065. It fixed auto-repeat processing for Safari.

**#29 - 06/06/2016 11:21 AM - Sergey Ivanovskiy**

- *File web-chui-safari.png added*

The CHUI web client displays the screen incorrectly in the case of Safari. The width and height of the canvas, font family name and size, font's width and height equal to the same values calculated for Chrome and Firefox: `width=560=7*80` and `height=312=13*24`, `font="12pt monospace"`, `font height = 13` and `font width = 7`, but the visual appearance of the Safari font looks different. Please look at the screen shot that describes this case.

**#30 - 06/07/2016 12:33 PM - Sergey Ivanovskiy**

Now working on the note 29 to detect the font height correctly using offscreen canvas. The tests to detect the first and the last valuable (white) pixel on the offscreen canvas are not passed.

**#31 - 06/07/2016 03:59 PM - Sergey Ivanovskiy**

Found an interesting Safari feature to draw vertical line `"\u2502"` on the offscreen canvas for the '12px monospace' font and the default line style it uses alpha channels only. Thus

the offscreen canvas image data contains these types `[0,0,0,*]` of pixels. Prepared simple algorithm that takes into account the mean and variance of horizontal pixels distribution in order to calculate the font height correctly.

**#32 - 06/07/2016 08:27 PM - Sergey Ivanovskiy**

- *File 3080\_clipboard\_11.txt added*

Sergey Ivanovskiy wrote:

Found an interesting Safari feature to draw vertical line `"\u2502"` on the offscreen canvas for the '12px monospace' font and the default line style it uses alpha channels only. Thus  
the offscreen canvas image data contains these types `[0,0,0,*]` of pixels.



This statement is not true. It seems that the test script code causes this confusion, but I am not sure.

Prepared simple algorithm that takes into account the mean and variance of horizontal pixels distribution in order to calculate the font height correctly.

Committed revision 11066 fixed the font detection and applied it for the Safari case. The vertical '\u2502' char has 10 pixels in its height for the "12px monospace" font, but the tested text string "\u2502Eghy" has 14 pixels due to the 'E' char. It needs to separate the vertical and horizontal border drawings in the case of the CHUI Web client and to remove the provided solution because it checks what text is in the given cell to detect the border case.

#### #33 - 06/08/2016 12:53 PM - Sergey Ivanovskiy

Greg, please review the committed revision 11068 (3084a). This task is ready.

#### #34 - 06/08/2016 02:47 PM - Greg Shah

Code Review Task Branch 3080a Revision 11068

Wow! That is a lot of work. Good job.

1. In p2j.screen.js, the fontMetrics constructor function uses

```
fontCanvas.height = height;
fontCanvas.width = width;
```

instead of the previous approach:

```
fontCanvas.setAttribute('height', height);
fontCanvas.setAttribute('width', width);
```

What do we lose or gain by this change?

2. Did you create the new font metrics approach from scratch? It is an interesting approach.
3. The old font metrics approach had a manual adjustment for windows platforms, which has been removed. Is that intentional?
4. For what platform + browser combinations have you tested the new font metrics approach?
5. In SystemAction, please split the class definition (public class SystemAction extends MouseWidgetAction) into 2 lines.
6. HandleCommon and HandleOps are each missing a history entry.
7. Please add javadoc to focusHiddenInputField() in p2j.clipboard.js.
8. Please cleanup the debug code and unnecessary commented code in p2j.screen.js.

**#35 - 06/08/2016 03:15 PM - Sergey Ivanovskiy**

1. In p2j.screen.js, the fontMetrics constructor function uses instead of the previous approach:  
What do we lose or gain by this change?

It is shorter and CanvasRenderer.prototype.resize already uses this short form. It works with IE, Firefox, Chrome and Safari.

2. Did you create the new font metrics approach from scratch? It is an interesting approach.

I use mean and variance approach because antialiasing transforms pixels so that they become colored in gray colors and to detect a char height it needs to take into account only valuable pixels. As an example, the vertical char '\u2502' has only 10 valuable pixels for '12px monospace' in Safari but the other pixels are faded. Otherwise, in Firefox '\u2502' has 13 valuable pixels.

3. The old font metrics approach had a manual adjustment for windows platforms, which has been removed. Is that intentional?

Yes.

4. For what platform + browser combinations have you tested the new font metrics approach?

I tested these changes with IE, Firefox, Chrome and Safari. Planning to rebase 3084a and to fix 5, 6, 7, 8.

**#36 - 06/08/2016 03:53 PM - Greg Shah**

I uses mean and variance approach because antialiasing transforms pixels so that they become colored in gray colors and to detect a char height it needs to take into account only valuable pixels. As an example, the vertical char '\u2502' has only 10 valuable pixels but the other pixels are faded.

Please add this information as a comment in the fontMetrics object. It really helps one understand why the approach was taken.

#### #37 - 06/08/2016 04:18 PM - Constantin Asofiei

Sergey, one question about 3084a: in p2j.js createKeyEvent, you have a finally block which uses an event object which may have been created in a catch block:

```
catch (ex2)
{
    event = document.createEvent("UIEvents");
}
```

This assumes that the UIEvents event will always be created, and never fail: why is this valid? Which browser does the catch target? If document.createEvent("UIEvents"); fails, won't event be undefined in the finally block?

#### #38 - 06/08/2016 04:38 PM - Sergey Ivanovskiy

I used here the code from YUI library <https://github.com/yui/yui3/blob/master/src/event-simulate/js/event-simulate.js>. It seems that it is a Safari use case.

#### #39 - 06/08/2016 04:43 PM - Sergey Ivanovskiy

Committed revision 11070 (3084a) fixed 5-8 and added comments to fontMetrics. 11069 revision is a rebase from the 11041 trunc revision. I tested these combinations:

IE11 and Window7 32-bit, Chrome, Firefox and Ubuntu 15.10, Safari and Mac OS X El Capitan 10.11.5.

#### #40 - 06/08/2016 05:09 PM - Sergey Ivanovskiy

- File GoogleChromeWin7-32bit.PNG added

I detected the same problem that appears using Safari on Google Chrome ver. 50.0.2661.102 and Win7 32-bit image. The issue is char '\u2502' is uniformly faded so it can be detected by the mean and variance rough method. It requires a more sensitive approach. Please look at the picture.

#### #41 - 06/08/2016 05:21 PM - Sergey Ivanovskiy

Zooming 4 times this picture helps to observe that a vertical char consists from 3 parallels and the middle line is lighter than its neighbors. Thus its height must be smaller than used char height to draw cells. It needs to investigate thoroughly.

#### #42 - 06/09/2016 09:22 AM - Greg Shah

Code Review Task Branch 3084a Revision 11070

The changes are good.

**#43 - 06/09/2016 09:24 AM - Greg Shah**

Constantin Asofiei wrote:

Sergey, one question about 3084a: in p2j.js createKeyEvent, you have a finally block which uses an event object which may have been created in a catch block:

[...]

This assumes that the UIEvents event will always be created, and never fail: why is this valid? Which browser does the catch target? If document.createEvent("UIEvents"); fails, won't event be undefined in the finally block?

I agree that this code should be written to be more explicitly safe. Even if it works reliably now, we are depending on certain browsers to always return something in the catch case, which may break over time as the browser support changes.

**#44 - 06/09/2016 09:32 AM - Sergey Ivanovskiy**

If the browser API is changed, then this function is failed now. It can be changed to return null object without throwing any exception. In any case we are failed with repeats keys in that case, because this function is used now for the Safari use case only.

**#45 - 06/09/2016 09:45 AM - Greg Shah**

If the browser API is changed, then this function is failed now. It can be changed to return null object without throwing any exception. In any case we are failed with repeats keys in that case, because this function is used now for the Safari use case only.

Agreed. Please go ahead with protection code in the finally to not throw an exception but to "fail gracefully". It will still fail, but readers of the code will understand that we have planned accordingly AND it will be easier to see the reason.

**#46 - 06/09/2016 12:50 PM - Sergey Ivanovskiy**

Sergey Ivanovskiy wrote:

I detected the same problem that appears using Safari on Google Chrome ver. 50.0.2661.102 and Win7 32-bit image. The issue is char '\u2502' is uniformly faded so it can be detected by the mean and variance rough method. It requires a more sensitive approach. Please look at the picture.

The last Chrome issue is related to that the cell width is not whole number and it causes these artifacts due to the different Canvas API implementations. I think it is not succeeded if we continue to draw chars to display boundaries.

We use `dx = ctx.measureText("\u2500").width;` to calculate the cell's width.

For the Chrome on Win7 - 32 bit the horizontal char '\u2500' width is 6.59765625 for the "12px monospace" font and horizontal chars are displayed on the canvas with gaps. The gaps also appear if we round this value to the closest integer 7. But for the Safari the horizontal char '\u2500' width has

7.201171875 pixels rounded to 7. And it gives the vertical gaps only (fixed by drawing two vertical chars), but without rounding we can observe horizontal and vertical gaps. The Chrome on Ubuntu 15.10 gives 7.224609375 and only the Firefox on Ubuntu 15.10 and IE11 on Win7-32bit both give the whole number 7. We can try to rework this approach if we add an attribute to each cell that describes a target cell is a boundary or not. It gives a possibility to use graphics instead of chars drawings.

Greg, would you prefer that this task should be done in the separate task or it should be done here?

**#47 - 06/09/2016 01:21 PM - Greg Shah**

would you prefer that this task should be done in the separate task or it should be done here?

Please create a new task. Please include zoomed screen-shots of each issue.

**#48 - 06/09/2016 09:29 PM - Sergey Ivanovskiy**

Committed revision 11071. It needs to rebase to 11042 trunc and to check menu functionalities.

**#49 - 06/10/2016 08:35 AM - Greg Shah**

Code Review Task Branch 3084a Revision 11072

The changes are good.

**#50 - 06/15/2016 04:20 PM - Greg Shah**

- Status changed from New to Closed
- % Done changed from 0 to 100

**#51 - 06/15/2016 04:24 PM - Greg Shah**

Task branch 3084a was merged to trunk as revision 11045 and that revision contains the changes that resolve this task.

**#52 - 11/16/2016 12:22 PM - Greg Shah**

- Target version changed from Milestone 16 to Cleanup and Stabilization for GUI

Files			
testKey.html	5.57 KB	05/31/2016	Sergey Ivanovskiy
demo_widgets_mac_os_x.png	208 KB	05/31/2016	Sergey Ivanovskiy
3080_clipboard_10.txt	21 KB	06/03/2016	Sergey Ivanovskiy
3080_clipboard_10.txt	8.22 KB	06/03/2016	Sergey Ivanovskiy
web-chui-safari.png	510 KB	06/06/2016	Sergey Ivanovskiy
3080_clipboard_11.txt	15.8 KB	06/08/2016	Sergey Ivanovskiy
GoogleChromeWin7-32bit.PNG	43.7 KB	06/08/2016	Sergey Ivanovskiy