**BugReport ID="2"**

**Title: "(449596) Firefox - remove the browser.sessionstore.enabledpref "**

That pref was thought to be for extensions which wanted to completely replace our own Session Restore functionality.

I'd much rather encourage extensions authors to override both nsSessionStartup and nsSessionStore to provide the same API with their own functionality (or implementing a dummy-API and making sure for themselves that they've correctly replaced all known consumers). That's what browser.sessionstore.max\_tabs\_undo is for: setting it to 0 effectively disables the feature. &gt; and others don't want any session tracks stored in memory at all. Then again, we don't save any data that wouldn't be in memory, anyway, or do we? So max\_tabs\_undo=0 + resume\_from\_crash=false is fine for now, until proper \"private browsing\" is supported. &gt;+ // XXXzenikoshould't we just disable this item as we disable &gt;+ // the tab browser-multiple items above - for consistency? &gt;+this.mUndoCloseTabMenuItem.hidden = &gt;+ Cc[\"@mozilla.org/browser/sessionstore;1\"].

&gt;+ getService(Ci.nsISessionStore).&gt;+ getClosedTabCount(window) == 0;+1,r=me, &gt; +1,That's bug 350731. Care to convince mconnor that he's wrong? ;-) Pushed as 17120:e712e96d7861 and 17121:adb1ef78dd21. &gt; I currently have only one problem , how to disable the restore after restart. You've got several options for that:\* Set the prefsbrowser.sessionstore.resume\_from\_crash and browser.sessionstore.resume\_session\_once both to false as early as possible and make sure that browser.startup.page isn't 3. \* Delete the file sessionstore.js as early as possible (e.g. when the profile-after-change notification is dispatched).\* For Firefox 3.1: Respond to the sessionstore-state-read notification by setting the subject's data member to an empty string (cf. bug 448741 comment#6).Or is there a use case I'm missing?

**BugReport ID="3"**

**Title :"(491925) Firefox - Disable multitouch \"rotate\" gesture for cycling tabs"**

I've noticed that I frequently trigger the rotate gesture accidentally, usually while scrolling.Gestures have improved since the original landing (when triggering the wrong gesture was really easy), but this articular gesture is still problematic. So, given this problem, we should just disable it for 3.5.[I'd also be open to tweaking it to make it much harder to trigger accidentally, dunno if that's possible.] See also bug 461376. And I also find that it's not just in Firefox, it's all over OSX. I wonder how much of this is us vs. odd timings causing misreads of the trackpad by OSX. That's probably because zoom is a subtle effect (for 1 step, relative to the distraction that an accidental tab switch causes). While playing with different browser.gesture.twist settings, I've found twisting through tabs often resulted in triggering a zoom. Probably because a twist is a very awkward gesture to make, especially if it's more than a quarter-turn. I've been using swipe left/right to change tabs and cmd-left/right to go back/forward, but I do have rotate as change tabs as well and I haven't accidentally triggered it. Maybe it's your rotating technique ;

It would be nice to leave the prefs visible for those who want to use them. Since this is now implemented on Win7 with Multitouch screens please be sure not to disable it for Windows. &gt; Since this is now implemented on Win7 with Multitouch screens please be sure not to disable it for Windows. Fine by me; though I'd be a bit wary of having Windows users hit the same problem with multitouch trackpads... And maybe Windows 7 is just better at discriminating between gestures. OTOH it wouldn't be a big deal to just disable it globally. Safari doesn't use it, and Rob confirmed that IE doesn't use it on Windows 7. Pinching is a natural gesture for page zoom (and more widely used/known), with a low penalty for accidentally triggering. I think most people would consider it a bug if we claimed to support multi touch, but pinch didn't do anything. Pinchy would've wanted it this way. &gt; This bug is just rotate, but do we want to consider pinching too? &gt; &gt; If we take those out, we'll have swipe up/down as jump to top/bottom and swipe left/right as back/forward by default. Pinch should definitely be left on for win7. Tracking is turned off so you get a nice zoom in/zoom out effect. Can we get the patch reviewed - I think we want to remove the rotate gesture on both Windows and OSX. Created an attachment (id=378424).both Windows and OSX. From update of attachment 378424 id. uir+r=beltzner. http://hg.mozilla.org/mozilla-central/rev/d19424342b43.Give empty commands for twist gestures, so they do nothing, but still show up as prefs in about:config. http://hg.mozilla.org/releases/mozilla-1.9.1/rev/3329a3997d7b

Verified fixed on trunk and 1.9.1 with builds on OS X and Windows like: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.2a1pre) Gecko/20090525 Minefield/3.6a1pre ID:20090525031110 Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.1pre) Gecko/20090526 Shiretoko/3.5pre ID:20090526031155 Off the back of this, would there be any chance of getting...browser.gesture.swipe.left.shift added and set to Browser:PrevTabbrowser.gesture.swipe.right.shift added and set to Browser:NextTab Having made this change, I must admit to now finding Firefox almost 'alien' to use on other multi-touch machines!

<https://bugzilla.mozilla.org/show_bug.cgi?id=502500>

**BugReport ID="7"**

**Title :"(437797) Firefox - about:mozilla page is hardcoded to LTR in RC2"**

The about:mozilla page in RC2 is right-aligned, but has LTR directionality instead of RTL. as a result, the dot at the end of the sentence is on the right side instead of the left, and both the parentheses around the \"10th edition\" line are on the right side. Same behavior also affect other internal about pages.\* about:plugins. pages that get RTL: \* about:crashes \* about:robots - Has some open issues. See bug 427029. Pages that not translated, such as about:cache, about:credits and about:buildconfig should stay LTR. theabout:mozilla page used to be a file in the language pack, allowing the localizer to control it's look. moving this out of the language pack created this problem. the file dom/chrome/global.dtd control has direction control with the entity locale.dir. "Because directionality is an integral part of the document structure, markup should be used to set the directionality for a document or chunk of information, or to identify places in the text where the Unicode bidirectional algorithm alone is insufficient to achieve desired directionality.\" To me, the trunk solution for this is to have a separate entity for each of those pages, to work around bugs like bug 427029. Once you localize a page we ship, you would then set the dir from ltr to rtl. That's not a solution for the 1.9 branch, though, there we should just re-use locale.dir from global.dtd. IMHO. Gavin, what do you think?In reply to comment #8) &gt; To me, the trunk solution for this is to have a separate entity for each of those pages, to work around bugs like bug 427029.&gt; Once you localize a page we ship, you would then set the dir from ltr to rtl. &gt; That's not a solution for the 1.9 branch, though, there we should just re-use locale.dir from global.dtd. Well, that depends on how disciplined folks are.We could make that call per page, like in the case of the robots page, it might make sense to have an open window for not translating it, for neterror pages probably less. There we could even get extreme and do it per div ;-). http://hg.mozilla.org/index.cgi/mozilla-central/rev/d12488eef3f5

**BugReport ID="9"**

**Title :"(328600) Firefox - Option for mailto: links to go to a web based e-mail client (webmail service)"**

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.12)Gecko/20050915 Firefox/1.0.7 (ax)

Build Identifier: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.12)Gecko/20050915 Firefox/1.0.7 (ax)

Firefox currently supports a wide number of stand alone e-mail applications (including it's own Thunderbird client), when an e-mail link is clicked on in Firefox. To the best of my knowledge it still does not support web based e-mail clients such as g-mail in this fashion. Other enhancement requests like this have been marked as dups of bug 11459, but I think that's wrong. As far as I can tell, bug 11459 is a combination of a design flaw specific to Seamonkey and a tangentially related enhancement request that also applies to Firefox, and nothing is happening there. Webmail services can help give users programs that alter system mailto: behavior (e.g. the unofficial \"G-Mailto\" and the official \"Gmail Notifier\"), and Firefox users can download the WebmailCompose extension, but i think it would be better if this were an option in Firefox itself. Using Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1) Gecko/20090615. Firefox/3.5. Is this not now fixed, Indeed.

**BugReport ID="32"**

**Title: "(238215) Firefox - Add a history window (like in Seamonkey)"**

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7b)Gecko/20040316

Build Identifier: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6)Gecko/20040206 Firefox/0.8 There needs to be a way to open sidebar resources, such as the history and bookmarks, in their own window. I like the CTRL-B and CTRL-H features of Mozilla and would like that carried over. I know bug 178199 discusses CTRL-B (WONTFIX but perhaps CTRL-SHIFT-B) in particular but I'm talking about depending on the side bar at all. Reproducible: Always Steps to Reproduce: A bookmark manager window already exists, and bug 178199 covers giving it a better keyboard shortcut than Alt,B,M. I don't see a bug for adding a history window. how about something like about:history? similar to about:config - just a full page tree with a search/filter field at the top.

you wouldn't really need a menu or toolbar buttons like with the bookmarks manager.

but like the bm and unlike the sidebar, it would have all the sortable columns: name, url, last visited, first visited, number of visits, etc. it would be the \"power-user\" supplement to the simple history sidebar. sure, the sidebar needs a few things fixed like searching urls, but other than that, i think it's fine for most uses - exactly what sidebars are for - a quick way to access something without leaving/obscuring the current page. and an about:history page would nicely fill in the gap left on the advanced end. a whole new history window is probably overkill, and a page could just as easily be opened in its own window or in a tab. extensions to open the history page and filter pages with \"extensions.\" compare:

chrome://browser/content/history/history-panel.xul chrome://browser/content/bookmarks/bookmarksPanel.xul

chrome://browser/content/bookmarks/bookmarksManager.xul

about:config, reporter, not sure if you're asking for a whole new history window, or just the ability to open the sidebar history in its own window. if the latter is the case, then just open the above chrome url in a new window. the component and other fields probably need to be updated.

I'm really looking for two things:

1. A separate window so I can keep going back to the same window and open multiple windows from this window. chrome://browser/content/history/history-panel.xul would meet this need.

2. Being able to see and sort the history.

Seamonkey offers tabs for Title, Location, Last Visited, First Visited, Hostname, Referrer, and Visit Count. Not all of this is needed but at least the first four are helpful. I'm really looking for two things: &gt; &gt; 1. A separate window so I can keep going back to the same window and open multiple windows from this window. &gt; chrome://browser/content/history/history-panel.xul would meet this need. &gt; &gt; 2. Being able to see and sort the history. Seamonkey offers tabs for Title, Location, Last Visited, First Visited, Hostname, Referrer, and Visit Count. &gt;

**BugReport ID="1"**

**Title: "(495584) Firefox - search suggestions passes wrong previous result to form history"**

MattN noticed a problem with the WIP patch from bug 469443 applied.  
When typing in the search box, sometimes search-suggestion entries would be displayed above the divider (where entries for previous matching searches are).The problem here is that nsSearchSuggestions.js is passing the wrong previousResult to form history. Instead of it being the previous form history search result, it's the SuggestAutoCompleteResult result (which contains the union of the form-history and search-suggest entries).  
So, when form history refines its results as you time, it can actually add \*more\* entries as data leaks from the suggestions result into form history result, and it thus looks like the divider is being drawn in the wrong place. startHistorySearch()'s previous result contains [\"blah\", \"baaa\", \"bloop\", \"bzzz\"], Satchel filters this down to [\"blah\", \"bloop\"] to match the new \"bl\" search string nsSearchSuggestions's onReadyState() change is called with updated search suggestions, builds up a new list of results, but sees that the form history result now has \*two\* entries. I suspect nsSearchSuggestions needs to throw away the old form history result when the search string changes like this, but I'm not entirely sure it's responsible for doing so, maybe satchel should be smarter about throwing away a previous result when the previous result's search string doesn't have a common prefix. So, there's a .\_formHistoryResult in the SuggestAutoCompleteResult wrapper (used to combine form history with search suggestions), and also a .\_formHistoryResult in SuggestAutoComplete (the service itself, used to hold onto a form history result until a search suggestion is available).The simple fix it to just discard the service's form history result copy when startSearch() is called with a null previous result.

**BugReport ID="10"**

**Title :"(156905) GIMP - GimpAspectPreview doesn\'t respect the layer offset and can crash"**

draw something on the drawable (using some colors)make a selection selecting only some parts of your drawing use the channel mixer.

The selection looks wrong in the plugin, and the plugin can even crash. Clearly what it is doing is bad, but what \*should\* it be doing? Should the preview follow the shape of the selection bounds, the image, or the drawable?

(1) Show the entire drawable in an aspect preview, and mask changes using the current selection.

(2) Show only the part of the drawable lying within the selection bounds. It isn't clear to me which of them is better.

Bumping to 2.4 since this is not likely to get fixed for 2.2. Bumping to Future since this is not likely to get fixed for 2.4. Setting back to the 2.4 milestone since it's a crash and should, if possible, be looked at before the 2.4 release (channel\_mixer:6701): LibGimpWidgets-CRITICAL \*\*: gimp\_preview\_area\_mask: assertion `mask != NULL' failed. Doesn't channel mixer use GimpZoomPreview (not GimpAspectPreview)? Seems like both GimpZoomPreview and GimpAspectPreview have problems when steps in reporter's comment are taken. This one is nasty to track down. The preview code then calls gimp\_drawable\_get\_sub\_thumbnail on the selection mask with the following coordinates: src\_x: 117, src\_y: 89, src\_w: 268, src\_h: 152. which unsurprisingly fails, since src\_x+src\_w is bigger than the selection mask width.

The source coordinates seem wrong to me and originate from gimp\_zoom\_preview\_get\_source\_area(), but I have not yet tracked this down. Created an attachment (id=117704) [details]. GimpZoomPreview handle layer offsets

This patch should fix GimpZoomPreview to handle layers with offsets with selections. Area displayed in preview is intersection of layer and selection. For GimpAspectPreview selection mask has to be applied for thumbnail of the whole drawable (selection can intersect partialy with drawable area).

**BugReport ID="11"**

**Title : "(164995) GIMP - Remember last scale method setting"**

When images are rescaled now the GIMP always defaults to linear scaling. Sometimes I want to do a bunch of images with cubic scaling.. The default interpolation routine can be choosen in the Preferences dialog. I just thought it would be easier if it just remembered the last setting, that way a newbie does not have to search trough the preferences, and it is handy if you want to scale a set of images in a different way as normally Jaap is right. This applies in general to options that the user can set without going into the preferences dialog: just remember the last setting, and there is no need for an entry in the preferences dialog.

Because you then still need an entry in preferences and you need an extra entry in the scale dialog whether to save the last used setting. Just always remembering the last setting is conceptually by far the easiest for new users. However, I disagree with the proposal that the last setting should automatically become the default: I have set the default to Cubic (I might change it to Lanczos soon) and I expect it to be used all the time. So if there is a way to change the default from the Scale dialog (which is probably a good idea), then it should require an explicit action from the user. I didn't know about the preferences setting until I saw this bug report.

\* app/dialogs/image-scale-dialog.[ch]: made the ImageScaleDialog struct private, return a GtkWidget\* from image\_scale\_dialog\_new() and use a GimpScaleCallback in the public API.

\* app/actions/actions-types.h: #include \"dialogs/dialogs-types.h\"

\* app/actions/image-commands.c: changed according to image scale dialog changes. Also remember the resize dialog's unit (if it is \"percent\").

\* app/actions/layers-commands.c: remember scale and resize dialog stuff as in image-commands.

**BugReport ID="12"**

**Title : "(170801) GIMP - Converting image from grayscale to black&white is painfully slow"**

1. Open a large grayscale image of your choice (e.g. larger than 2000\*2000 pixels, maybe a scan result from your scanner)

2. Use \"Tools/Color Tools/Threshold\" to apply some threshold choosen arbitrarily.

This tool is pretty fast.

3. Now you have a 8bit grayscale image, which actually consists only of color values \"0\" and color values \"255\". Converting it to a 1bit black&white image should be a very fast operation (replacing 0 (8bit) by 0 (1bit) and replacing 255 (8bit) by 1 (1bit)), but it is not:

4. Choose \"Image/Mode/Indexed...\", and there:

4.1 Colormap: \"Use black and white (1-bit) palette)\" (this should be the fastest option possible)

4.2 Dithering: \"none\" (this should be the fastest option possible)

4.3 Now hit \"OK\"

5. The conversion will work, but it will take many seconds.

Image where to 1bit conversion is either slow or buggy (with GIMP 2.2.4). That's odd. I just tried to use your suggested workaround and was happy that the conversion was as fast one should expect. However, the conversion was plainly wrong, it produced an image which was just black (the input image was nearly white). Do following on the atached image: 1. Choose \"Image/Mode/Indexed...\", and there: 1.1 Colormap: optimal palette (wether 2 oder 256 colors, it does not matter)

1.2 Dithering: \"none\" (but I think it does not matter). Interesting -- clearly the generation of an optimal palette is not done correctly for this image. In any case, I've found a solution to the original problem, I think. I found a very simple change that gives a major speedup when converting grayscale images to the mono palette. It's now at least as fast as the Threshold tool, so I'm going to resolve this bug as FIXED. I'd be pleased if you used my test case image. If you can't attach it then what revision of gimpimage-convert.c contains the fix? Revision 1.156. Here is the patch: --- gimpimage-convert.c 27 Feb 2005 14:03:50 -0000 1.155; +++ gimpimage-convert.c 18 Mar 2005 17:34:34 -0000 1.156; @@ -4057,7 +4057,6 @@ initialize\_median\_cut (GimpImageBaseType; if (palette\_type == GIMP\_WEB\_PALETTE ||; - palette\_type == GIMP\_MONO\_PALETTE ||; palette\_type == GIMP\_CUSTOM\_PALETTE); {; switch (dither\_type). The 'mono' palette option doesn't even bother to start this pre-pass because it could only possibly pay off the extra effort if the entire image is pure black and pure white, which is expected to be a comparatively rare occurance. The 'mono' palette option doesn't even bother to start this pre-pass because it could only possibly pay off the extra effort if the entire image is pure black and pure white, which is expected to be a comparatively rare occurance. If you mean GIMP's internal memory requirements you're wasting your time -- all Indexed images are 8/16bpp internally (the same as greyscale images), so you're not saving any memory (instead you may be using quite a lot more, for various I mean OpenOffice's internal memory requirements.

**BugReport ID="13"**

**Title : "(364852) GIMP - screenshot plug-in incomplete for Windows platform"**

Please describe the problem: When going to File-&gt;Acquire-&gt;Screenshot in the Windows port of the development build GIMP 2.3.11, firstly there are two different entries \"Screen Shot\" and \"Screenshot\" and secondly, functionality to grab only a portion of the screen is broken. 1. Launch GIMP 2.3.11 Development for Windows on a Windows 2000 platform.

2. Got o File-&gt;Acquire-&gt;Screenshot. (Note the two different entries for \"Screen Shot\" and \"Screenshot\".)

3. Tick \"Select a Region to Grab.\"

4. Set the delay.

5. Click \"Grab.\"

Actual results: The whole screen is captured each time. Expected results: The ability to select a portion of the screen would be available, probably via mouse click-and-drag functionality. We know that the port of the screenshot plug-in for Windows is incomplete. If this is the case, perhaps it is a very good idea to drop support for Win32 and consolidate on making sure the \*Nix implementation is as good as possible. Supporting Win32 is probably just draining more resources away. IIRC Tor is working on this. apf, you are using a development version. Tor said that it is impossible to implement the look and feel of the X11 screenshot plug-in on the Windows platform. He suggested that we stick with Winsnap for the Win32. 2006-10-27 Sven Neumann &lt;sven@gimp.org&gt; \* configure.in; \* plug-ins/common/plugin-defs.pl; \* plug-ins/common/Makefile.am: reverted changes from 2006-04-06 and build screenshot plug-in optionally again.