

JavaScriptDebuggerWindowsFunctionalSpec

Table of Contents »

The NetBeans JavaScript Debugger

Debugger Windows

Breakpoints and the Breakpoints Window

Sessions Window

Sources Window

Threads (Windows and Call Stack) Window

Call Stack Window

Local Variables Window

Watches Window

Output Window

Runtime DOM Window

CSS Window

Network Monitor Window

Future Features

Profile Window

Network Activity Window

Sources Window

CSS Window

The NetBeans JavaScript Debugger

Debugger Windows

The following is a description of the windows that are shown when a debug session is initiated. It does not define editor, navigator, outline, etc. In the description below sometimes I refer to a current context. Generally this context is only loaded when a break point is hit or as the user steps through the javascript source. In other words, Watches and Local Variables Window will not update it's information unless the rendering is in a paused state. This state is generally initiated by a breakpoint in the source, giving control back to the user.

Breakpoints and the Breakpoints Window

- Breakpoints are typically created by a user.

1. Adding a breakpoint via the editor
 2. Adding a breakpoint via the Breakpoint View
- A Breakpoint can be enabled or disabled via a checkbox in Breakpoint view.
 - Any given breakpoint is customizable meaning that the user can provide an expression that if evaluated true at the time of execution will trigger execution to pause.
 - The user can request a breakpoint when an error/exception is thrown (?)
 - As the JavaScript executes, the user will be notified when the breakpoint is hit. The line will be highlighted in the editor and the JavaScript will pause from its running state until the user decides how he or she wishes to proceed. The Watches Window, Local Variables Window, Thread and Call Stack Window will be fully populated at this time.

Context Menu Actions

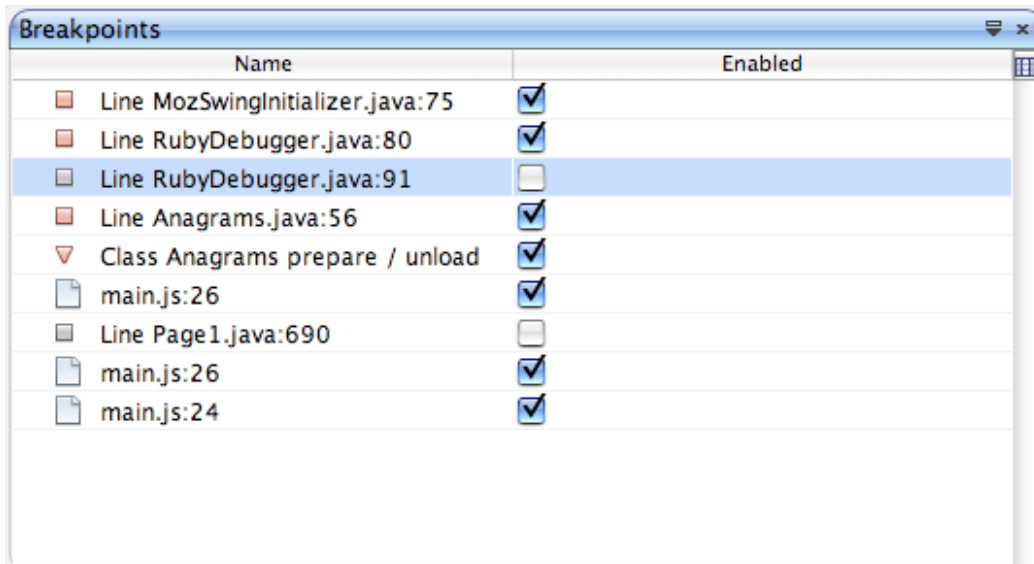
1. Go To Source
2. -----
3. Enable
4. Delete
5. Set Group Name
6. -----
7. New Breakpoint ...
8. -----
9. Enable All
10. Disable All
11. Delete All
12. -----
13. Customize^[1]
14. List Options (Sort, Change Visible Columns...)
15. -----
16. Refresh^[2]

Window Columns

1. Name
2. Enabled (Checkbox)

Actions

- Enable and disable break point by click on checkbox.
- Delete key will delete the selected breakpoint.



[#1]Customize option will be completed at a later date.

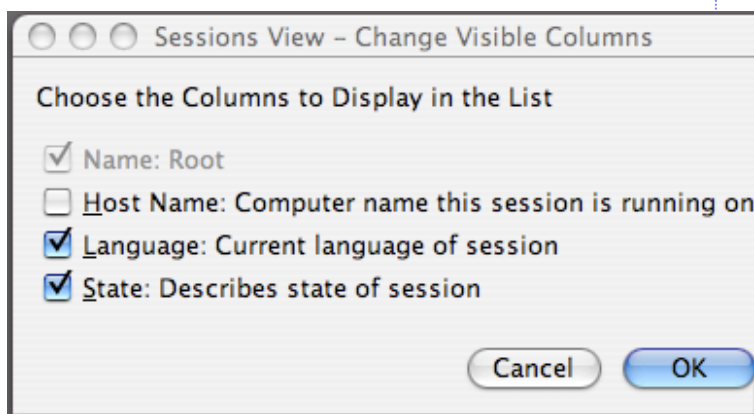
[#2]Refresh is necessary because once an expression is evaluated, it will not necessarily re-evaluate if the user is not currently stepping through the source.

Sessions Window

- Client Side Runtime will be represented by a session in Sessions View. This view will not be visible by default unless there are multiple sessions.

Context Menu Actions

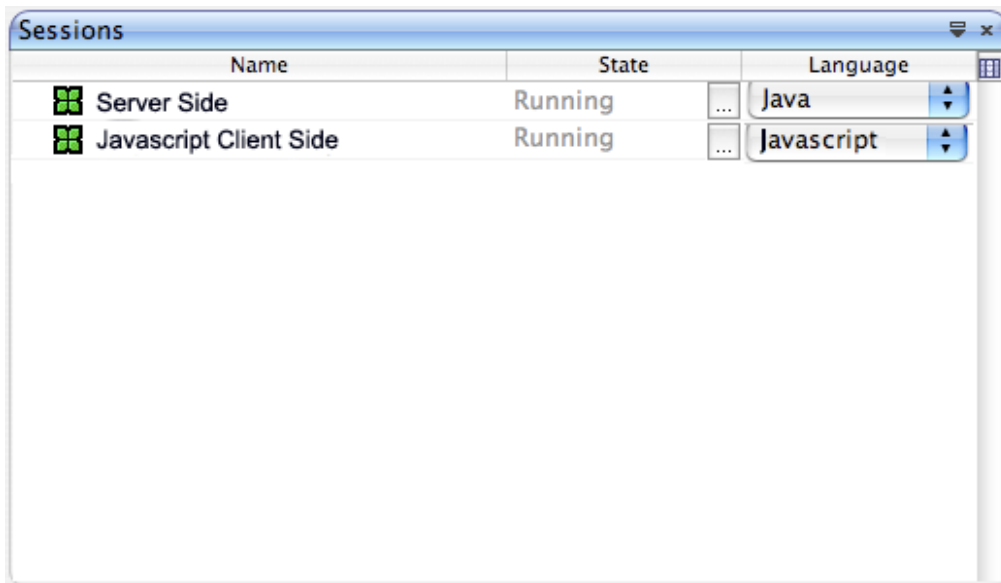
- Always
 - Finish All
 - List Options
- Column -> Move Left/Right (for a given column)
 - Sort -> None, Window Column Names
 - Change Visible Columns... - Opens the following dialog box.
- When Session Selected
 - Scope



- Debug All Threads
 - Debug Current Threads
2. Language
 3. Make Current (Disabled when only one Session is running)
 4. Finish

Window Columns

1. Name (Session Name)
2. State
3. Language (Drop Down)



Sources Window

- This Window displays all the sources related to current runtime context. The sources may be shown in the form of a URL (for remote resources) or a file path (for resources in the Project).
- From this view, the user can double click on a given source and it will open its relative source file (if it is one that we uploaded to the web server). Otherwise, Netbeans will grab the sources on the network and open it in a new file in the editor in a read-only state for the user to view.

Context Menu Actions

1. List Options
 - Sort
 - None
 - Source Root / Filter

- Use For Debugging
- Change Visible Columns

2. Go To Source

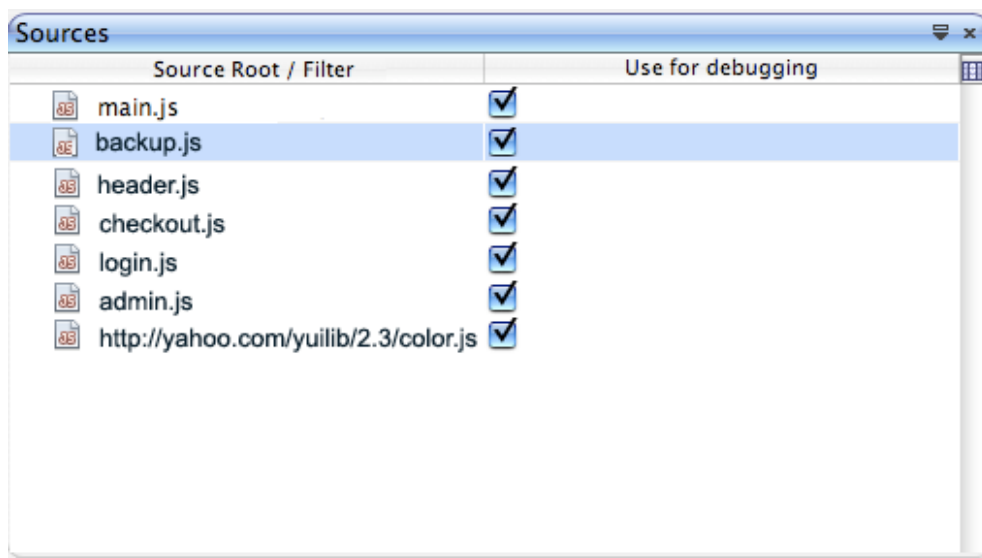
Note: Do we need "Add Source Root"?

Window Columns

1. File Name / URL
2. Use For Debugging (Checkbox - default true)

Actions

1. Double Click will trigger go to source.
2. Click on Checkbox will remove the sources from debugging.
3. Click on Column headers will modify Sorting (Ascending and Descending)



Threads (Windows and Call Stack) Window

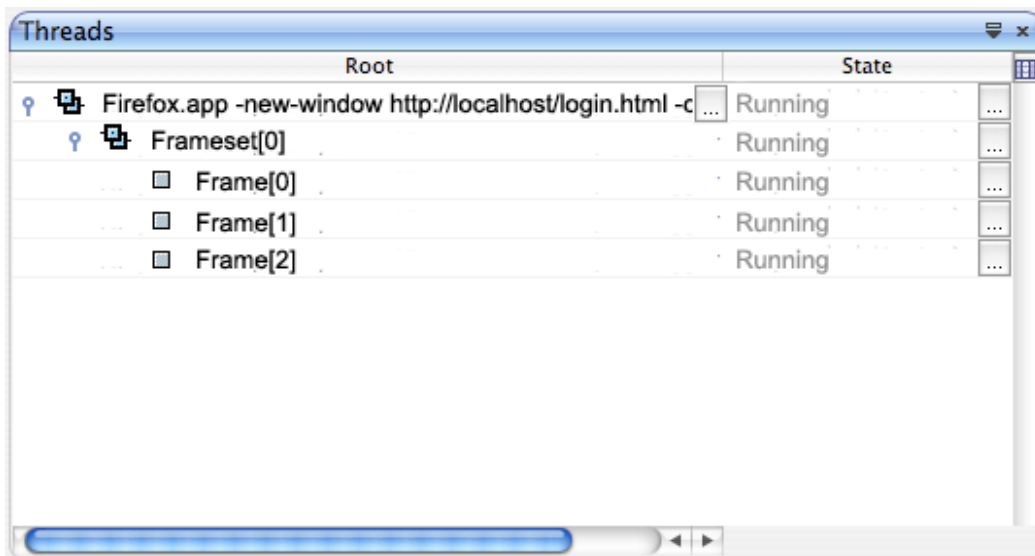
- In JavaScript there is not really a notion of Threads, however when there are multiple frames in a FrameSet or even a spawned window you are have distinct context in each. Therefore, we have decided to show top level Windows as the main(root) elements in the Thread view. The frames and iframes will be shown in their logical structure. The user can then switch between contexts depending on which window they select if the JavaScript execution is paused. (This may be inconsistent with NetBeans separate Threads and Call stack windows - HIE input required).
- Below each root is the relative call stack.
- For each element in the CallStack, the user can double click and it will be navigated to that portion in the source.

Context Menu Actions

- List Options

Window Columns

1. Root
2. State



Call Stack Window

- To be consistent with consistent with the NetBeans debugger UI the call stack window will show the call stack of the current context with the functionality described above (HIE input required). The Call Stack window will show the current window context information.

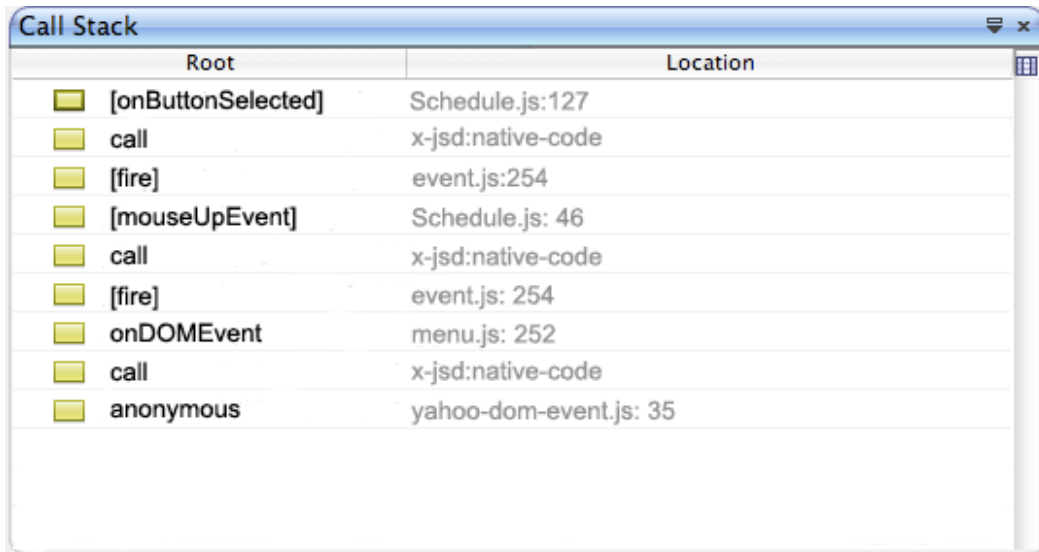
Context Menu Actions

- List Options (Sort, Change Visible Columns...)

Window Columns

1. Root
2. Location (Filename and Line)

Note: This probably needs to be revisited. Adding the location as default breaks away from the normative use of this window.



Local Variables Window

- This window displays all variables (or relative object properties) and their values known to the current context.
- Values are added and removed from the window as the context changes (or as the user steps through the javascript sources).
- The Local Variables window will show the current window + current call stack context information.

Context Menu Actions

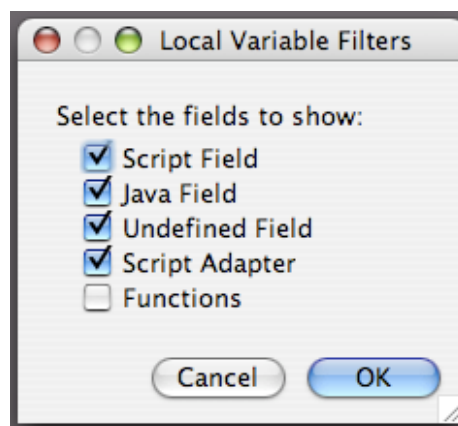
1. Filters
2. List Options (Sort, Change Visible Columns...)

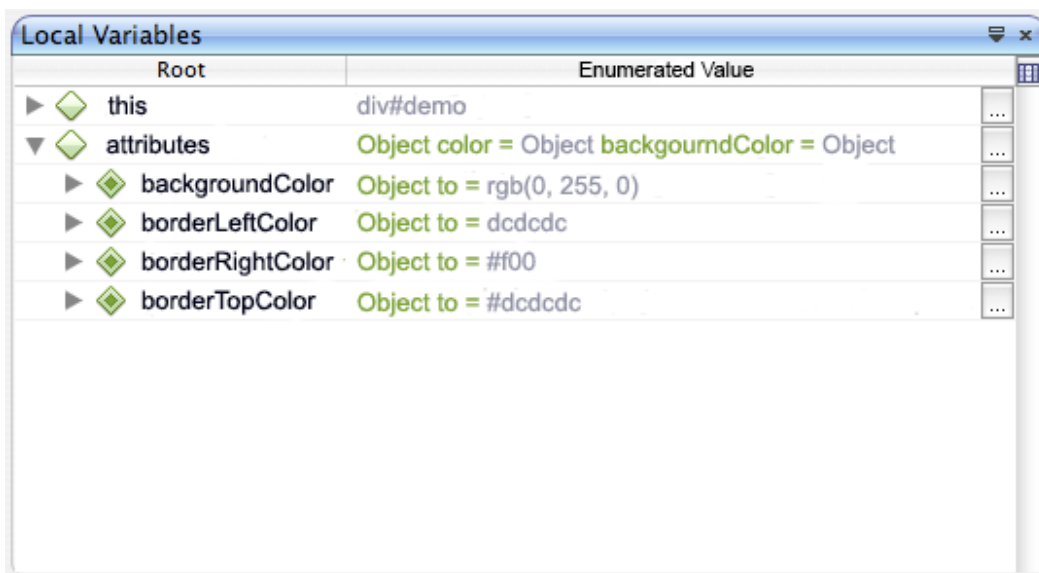
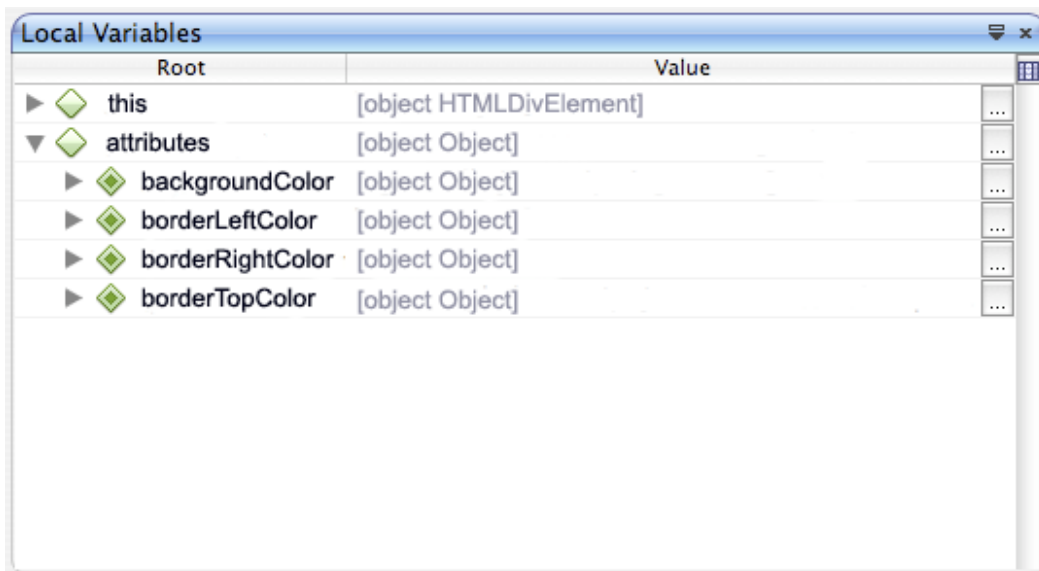
Window Columns

1. Root
2. Value
3. Enumerated Value^[1]

Note: I purposefully removed **TYPE** from visible column because it is not necessary in

JavaScript. Everything is an Object of some sort. **Local Variables Window** I have included to sample windows here. The first image shows the a column with the Javascript evaluated value while the second evaluates "to" on each variables or shows the enumerated values of a composite object.





[#1] Enumerated Values is probably not an appropriate title as when the item is not a composite object it evaluates ("to" and not "toString") on the given variable.

Watches Window

- This window provides developers a means of inspecting a text expression in any given context. When a user adds a watch, the evaluated expression and it's value is shown. This functionality is similar to the Netbeans Java Debugger.
- Scope: This expression is maintained beyond context and session (at the project level) and is only removed upon user request.
- Out of Bounds: If the JavaScript expression as provided by user cannot be evaluated in its given context it will simply state undefined.

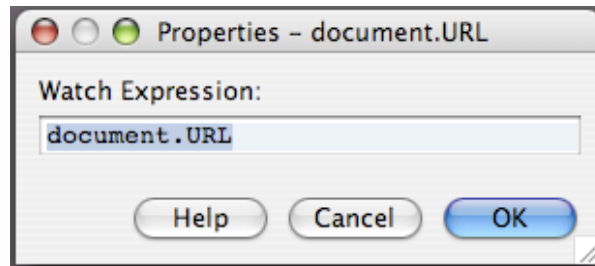
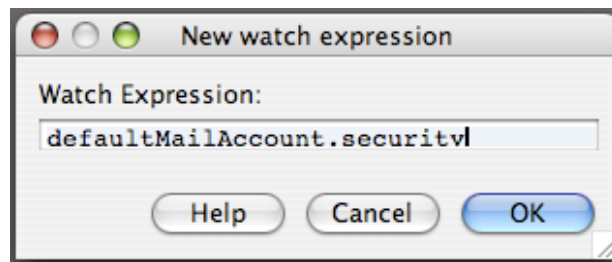
Context Menu Actions

1. New Watch ...
2. -----
3. Delete
4. Delete All
5. -----
6. Properties
7. List Options (Sort,
Change Visible Columns)

Window Columns

1. Root
2. Value (no value yet...)

Note: I purposefully removed **TYPE** from visible column because it is not necessary in JavaScript. Everything is an Object of some sort. **Actions** Delete key should delete a watch value.



Watches		
Root	Value	
scheduleDate	(undefined)	...
lastName	(undefined)	...
defaultMailAccount.name	INSERT_ACCOUNT_NAME_HERE	...
document.URL	http://localhost:8080/main.html	...

Output Window

- This window is similar to the JavaScript Error Console. It shows evaluation errors, thrown exceptions and other information sent to the console.

Runtime DOM Window

- The current DOM as parsed by the browser. This DOM can be modified dynamically

without editing source code.

- Their view is refreshed at
 - breakpoint
 - on demand
 - automatic (?)

CSS Window

- This view shows the current CSS state for any given DOM element. The view will also show how the cascading is happening. (Inspecting?)
- From this window you can edit a css element dynamically, however this data will reset when the page is refreshed.

Network Monitor Window

- Each request is represented by a line. There are four columns in each row (and the total table is sortable by any of these columns)
 1. request string (ie getModel?make=honda)
 2. domain string (ie mail.yahoo.com)
 3. request time
 4. initiating time of request
- As you mouse over each line, the tooltip will show the full url request.
- Each request can be expanded to show additional data information.
 - XMLHttpRequest GET will show the following.
 - Params: Name and value pairs of the URL
 - Headers: Package Headers of the Request and Response
 - Response: The raw text actually received in response to the request. (In the future, we may be able to do something special if we recognize a JSON object i the response.)
 - XMLHttpRequest POST will show the three above as well as the POST name value pairs.
- There will be 7 views on this window that act as a filter to viewable network activity.
 1. HTML - shows requests for HTML documents
 2. CSS - shows requests for CSS documents
 3. JS - shows requests for Javascript files
 4. XMLHttpRequest - shows AJAX request made
 5. Images - shows request make for images.

6. Flash - shows Flash requests.
7. ALL - show all network activity without any filtering.

Future Features

Profile Window

- This window will keep track of how long it takes for any given function to complete. By default, the methods that take the most time will be moved to the top of the list. There is a filtering mechanism in which the user may search for a Object name. The user can also sort by Object name and completion time.
- By default the profiler is turned off, but once turned on it can run in default mode or in a user specified context where it defines specific Object or Functions to listen to.

Network Activity Window

- The ability to rerun a request by simply double clicking on a request. This is usefully when the developer modifies server side code and would like to see the current response. However, it is not guaranteed that any Object properties will be updated.






Sources Window

- This sources window might also show the time it took to load this source file and it's size (?)

CSS Window

- The user should be allowed to press a commit button that will attempt to write the updated css values to the appropriate styles documents.

Attachments

BreakpointsWindow.png		29175 bytes
CallStackWindow.png		27499 bytes
LocalVariablesFiltersDialog.png		22618 bytes
LocalVariablesType1.png		25194 bytes
LocalVariablesType2.png		31694 bytes
NewWatchDialog.png		18697 bytes
PropertiesDialog.png		18636 bytes
SessionWindow.png		19156 bytes
SessionsViewChangeVisibleColumns.png		26877 bytes

[ThreadsWindow.png](#)
[WatchesWindow.png](#)
[netbugs.gif](#)
[netbugs.png](#)
[small_wallpaper1.png](#)
[sources.png](#)



22144 bytes
20626 bytes
297813 bytes
14046 bytes
20689 bytes
23629 bytes