

# GREASEMONKEY HACKS™

*Tips & Tools for Remixing  
the Web with Firefox*



O'REILLY®

**Mark Pilgrim**  
*Foreword by Aaron Boodman,  
Creator and Lead Developer, Greasemonkey*

HACK  
#66

## Add Saved Searches to Gmail

Keep often-used searches at your fingertips.

Gmail is Google's web mail application. In addition to the large amount of space that it provides, the main thing that sets it apart from the competition is the fact that its user interface is search-driven. It is therefore unfortunate that you must retype searches that you perform frequently. Many client-side email applications, such as Mozilla's Thunderbird, Gnome's Evolution, and Apple's Mail allow saved searches (also known as *persistent searches* or *smart folders*). This hack adds a similar feature to Gmail.

### The Code

This user script runs on the Gmail domain only. Initialization is rather complex, since the hack must create its own Gmail sidebar module. To make it easier to match the appearance of our sidebar with the rest of the Gmail interface, we use the CSS rules array to create a consistent set of CSS rules that we can reference later.

Each saved search is represented by a `PersistentSearch` object. Since searches must be saved across sessions, each object can be serialized to and deserialized from a string that we can then use with `GM_getValue` and `GM_setValue`. Additionally, each search can display how many results match it. To accomplish this, we use an `XMLHttpRequest` object to actually invoke the search URL, and then we parse the number of results from the response text. We cache the number of results to minimize hits on the Gmail server. Finally, we execute the search by calling Gmail's own `_MH_OnSearch` method.

To support editing of saved searches, we must override the main Gmail display and show our own interface instead. We must also do our own event handling, to deal with clicks on form buttons and other events. To make saved searches even more useful, we add some additional search operators that the user can enter, such as `after:oneweekago`. These are dynamically converted to absolute dates when the search is executed.

Save the following user script as `gmailsavedsearches.user.js`:

```
// ==UserScript==
// @name      Gmail Saved Searches
// @namespace http://persistent.info/greasemonkey
// @description Adds persistent searches to Gmail
// @include  http*://mail.google.com/*
// ==/UserScript==

// based on code by Mihai Parparita
// and included here with his gracious permission
```

```
// Utility functions
function getObjectMethodClosure(object, method) {
  return function() {
    return object[method].apply(object, arguments);
  }
}

function getDateString(date) {
  return date.getFullYear() + "/" +
    (date.getMonth() + 1) + "/" +
    date.getDate();
}

// Shorthand
var newNode = getObjectMethodClosure(document, "createElement");
var newText = getObjectMethodClosure(document, "createTextNode");
var getNode = getObjectMethodClosure(document, "getElementById");

// Contants
const RULES = new Array(
  // Block in sidebar
  ".searchesBlock {-moz-border-radius: 5px; background: #fad163; margin:
20px 7px 0 0; padding: 3px;}",
  ".refreshButton {display: block; cursor: pointer; float: right; margin-
top: -2px;}",
  ".searchesBlockList {background: white;}",
  ".listItem {color: #ca9c22;}",
  ".editLink {text-align: right; color: #ca9c22; padding: 2px 5px 5px 0;}",

  // Edit page
  ".searchesContainer {-moz-border-radius: 10px; background: #fad163;
padding: 10px;}",
  ".innerContainer {background: #fff7d7; text-align: center; padding:
10px;}",
  ".searchesList {width: 100%;}",
  ".searchesList th {text-align: left; font-size: 90%;}",
  ".searchesList td {padding: 10px 0 10px 0; vertical-align: bottom;}",
  ".searchesList td.divider {background: #fad163; height: 3px; padding:
0;}",
  ".editItem {font-size: 80%;}",
  ".labelCell {width: 210px;}",
  ".labelCell input {width: 200px;}",
  ".cancelButton {margin-right: 5px;}",
  ".editCell {}",
  ".editCell input {width: 100%;}",
  ".saveButton {margin-left: 5px; font-weight: bold;}");

const REFRESH_IMAGE = "data:image/
gif;base64,R0lGODlhDQAPANU5AM%2BtUs6sUunDX" +
  "PffPYt65WK%2BTRaiMQvXNYfDJX9m1VtSxVIBrM7GURsKiTZqBPeS%2Fwo940ZmApebBW6WK"
+
  "QbiaSd0wU35qMpV904t0N4NuNI120IFsM9u3V7mbSaaLQtazVcyqUZ6EP%2BC7WX1oMbuds"
+

```

```

    "semT62QRPjPYuvFXXtmMbSXR%2BK9WohyNvLKYOfBXPPLYJB40b6fS5R80%2B3GXqGGQK%2"
+
    "BSRauPROG9Ww5cK%2F%2F%2FwAAAAAAAAAAAAAAAAAAAAACH5BAEAADkALAAAAANA"
+
    "A8AAAZvwJxwSMzdiKcAg8YIDEyG4QPjABAUhgUuInQtAsQaDqcRwj7EUmY8yiUuReJtQInF"
+
    "h5JEAXQX3mwzD305FSRGBAN3Eys5HWM4LAddIiFCcmMbAkMcMghCBDgpEAUNKg4eL0MoFgI"
+
    "tAA0AnkQHmonBADs%3D";

const RESULT_SIZE_RE = /D\([\\"ts",(\d+),(\d+),(\d+),/;

const DEFAULT_SEARCHES = {
  "has:attachment": "Attachments",
  "after:today": "Today",
  "after:oneweekago": "Last Week"
};

const SEARCHES_PREF = "PersistentSearches";
const SEARCHES_COLLAPSED_PREF = "PersistentSearchesCollapsedCookie";

const ONE_DAY = 24 * 60 * 60 * 1000;

// Globals
var searches = new Array();
var searchesBlock = null;
var searchesBlockHeader = null;
var triangleImage = null;
var searchesBlockList = null;
var editLink = null;

var hiddenNodes = null;
var searchesContainer = null;
var searchesList = null;

function initializePersistentSearches() {
  var labelsBlock = getNode("nb_0");

  if (!labelsBlock) {
    return;
  }

  searchesBlock = newNode("div");
  searchesBlock.id = "nb_9";
  searchesBlock.className = "searchesBlock";

  // header
  searchesBlockHeader = newNode("div");
  searchesBlockHeader.className = "s h";
  searchesBlock.appendChild(searchesBlockHeader);

  var refreshButton = newNode("img");
  refreshButton.src = REFRESH_IMAGE;

```

```
refreshButton.className = "refreshButton";
refreshButton.width = 13;
refreshButton.height = 15;
refreshButton.addEventListener('click', refreshPersistentSearches, true);
searchesBlockHeader.appendChild(refreshButton);

triangleImage = newNode("img");
triangleImage.src = "/mail/images/opentriangle.gif";
triangleImage.width = 11;
triangleImage.height = 11;
triangleImage.addEventListener('click', togglePersistentSearches, true);
searchesBlockHeader.appendChild(triangleImage);

var searchesText = newNode("span");
searchesText.appendChild(newText(" Searches"));
searchesText.addEventListener('click', togglePersistentSearches, true);
searchesBlockHeader.appendChild(searchesText);

// searches list
searchesBlockList = newNode("div");
searchesBlockList.className = "searchesBlockList";
searchesBlock.appendChild(searchesBlockList);

editLink = newNode("div");
editLink.appendChild(newText("Edit searches"));
editLink.className = "lk cs editLink";
editLink.addEventListener('click', editPersistentSearches, true);
searchesBlockList.appendChild(editLink);

if (GM_getValue(SEARCHES_PREF)) {
  restorePersistentSearches();
} else {
  for (var query in DEFAULT_SEARCHES) {
    addPersistentSearch(new PersistentSearch(query, DEFAULT_
SEARCHES[query]));
  }
}

insertSearchesBlock();

if (GM_getValue(SEARCHES_COLLAPSED_PREF) == "1") {
  togglePersistentSearches();
}

checkSearchesBlockParent();
}

function refreshPersistentSearches() {
  for (var i=0; i < searches.length; i++) {
    searches[i].getResultSize(true);
  }
}
```

```
    return false;
  }

function insertSearchesBlock() {
  var labelsBlock = getNode("nb_0");

  if (!labelsBlock) {
    return;
  }

  getNode("nav").insertBefore(searchesBlock, labelsBlock.nextSibling);
}

// For some reason, when moving back to the Inbox after viewing a message,
// we seem to get removed from the nav section, so we have to add ourselves
// back. This only happens if we're a child of the "nav" div, and nowhere
// else (but that's the place where we're supposed to go, so we have no
// choice)
function checkSearchesBlockParent() {
  if (searchesBlock.parentNode != getNode("nav")) {
    insertSearchesBlock();
  }

  window.setTimeout(checkSearchesBlockParent, 200);
}

function restorePersistentSearches() {
  var serializedSearches = GM_getValue(SEARCHES_PREF).split("|");

  for (var i=0; i < serializedSearches.length; i++) {
    var search = PersistentSearch.prototype.
    fromString(serializedSearches[i]);

    addPersistentSearch(search);
  }
}

function savePersistentSearches() {
  var serializedSearches = new Array();

  for (var i=0; i < searches.length; i++) {
    serializedSearches.push(searches[i].toString());
  }

  GM_setValue(SEARCHES_PREF, serializedSearches.join("|"));
}

function clearPersistentSearches() {
  for (var i=0; i < searches.length; i++) {
    var item = searches[i].getListitem();
    if (item.parentNode) {
      item.parentNode.removeChild(item);
    }
  }
}
```

```
    searches = new Array();
  }

function addPersistentSearch(search) {
    searches.push(search);
    searchesBlockList.insertBefore(search.getListItem(), editLink);

    savePersistentSearches();
}

function editPersistentSearches(event) {
    var container = getNode("co");

    hiddenNodes = new Array();

    for (var i = container.firstChild; i; i = i.nextSibling) {
        hiddenNodes.push(i);
        i.style.display = "none";
    }

    searchesContainer = newNode("div");
    searchesContainer.className = "searchesContainer";
    searchesContainer.innerHTML += "<b>Persistent Searches</b>";

    container.appendChild(searchesContainer);

    var innerContainer = newNode("div");
    innerContainer.className = "innerContainer";
    innerContainer.innerHTML +=
        '<p>Use <a href="http://mail.google.com/support/bin/answer.py?answer=7190" target="_blank">operators</a> ' +
        'to specify queries. <code>today</code>, <code>yesterday</code> and <code>oneweekago</code> ' +
        'are also supported as values for the <code>before:</code> and <code>after:</code> ' +
        'operators. Delete an item\'s query to remove it.</p>';
    searchesContainer.appendChild(innerContainer);

    searchesList = newNode("table");
    searchesList.className = "searchesList";
    innerContainer.appendChild(searchesList);

    var headerRow = newNode("tr");
    searchesList.appendChild(headerRow);
    headerRow.appendChild(newNode("th")).appendChild(newText("Label"));
    headerRow.appendChild(newNode("th")).appendChild(newText("Query"));

    for (var i=0; i < searches.length; i++) {
        searchesList.appendChild(searches[i].getEditItem());

        var dividerRow = newNode("tr");
        var dividerCell = dividerRow.appendChild(newNode("td"));
        dividerCell.className = "divider";
        dividerCell.colSpan = 3;
    }
}
```

```
    searchesList.appendChild(dividerRow);
  }

  var newSearch = new PersistentSearch("", "");
  var newItem = newSearch.getEditItem();
  newItem.firstChild.innerHTML =
    "<h4>Create a new persistent search:</h4>" +
    newItem.firstChild.innerHTML;

  searchesList.appendChild(newItem);

  var cancelButton = newNode("button");
  cancelButton.appendChild(newText("Cancel"));
  cancelButton.className = "cancelButton";
  cancelButton.addEventListener('click', cancelEditPersistentSearches,
true);
  innerContainer.appendChild(cancelButton);

  var saveButton = newNode("button");
  saveButton.appendChild(newText("Save Changes"));
  saveButton.className = "saveButton";
  saveButton.addEventListener('click', saveEditPersistentSeaches, true);
  innerContainer.appendChild(saveButton);

  // Make clicks outside the edit area hide it
  getNode("nav").addEventListener('click', cancelEditPersistentSearches,
true);

  // Since we're in a child of the "nav" element, the above handler will get
  // triggered immediately unless we stop this event from propagating
  event.stopPropagation();

  return false;
}

function cancelEditPersistentSearches() {
  searchesContainer.parentNode.removeChild(searchesContainer);
  searchesContainer = null;

  for (var i=0; i < hiddenNodes.length; i++) {
    hiddenNodes[i].style.display = "";
  }
  getNode("nav").removeEventListener('click', cancelEditPersistentSearches,
true);

  return true;
}

function saveEditPersistentSeaches() {
  clearPersistentSearches();

  for (var row = searchesList.firstChild; row; row = row.nextSibling) {
```

```
var cells = row.getElementsByTagName("td");
if (cells.length != 2) {
    continue;
}
var label = cells[0].getElementsByTagName("input")[0].value;
var query = cells[1].getElementsByTagName("input")[0].value;

if (label && query) {
    var search = new PersistentSearch(query, label);

    addPersistentSearch(search);
}
}

// cancelling just hides everything, which is what we want to do
cancelEditPersistentSearches();
}

function togglePersistentSearches() {
    if (searchesBlockList.style.display == "none") {
        searchesBlockList.style.display = "";
        triangleImage.src = "/mail/images/opentriangle.gif";
        GM_setValue(SEARCHES_COLLAPSED_PREF, "0");
    } else {
        searchesBlockList.style.display = "none";
        triangleImage.src = "/mail/images/triangle.gif";
        GM_setValue(SEARCHES_COLLAPSED_PREF, "1");
    }
}

return false;
}

function PersistentSearch(query, label) {
    this.query = query;
    this.label = label;

    this.totalResults = -1;
    this.unreadResults = -1;

    this.listItem = null;
    this.editItem = null;
    this.resultSizeItem = null;
}

PersistentSearch.prototype.toString = function() {
    var serialized = new Array();

    for (var property in this) {
        if (typeof(this[property]) != "function" &&
            typeof(this[property]) != "object") {
            serialized.push(property + "=" + this[property]);
        }
    }
}
```

```
    }  
    return serialized.join("&");  
  }  
  
  PersistentSearch.prototype.fromString = function(serialized) {  
    var properties = serialized.split("&");  
  
    var search = new PersistentSearch("", "");  
  
    for (var i=0; i < properties.length; i++) {  
      var keyValue = properties[i].split("=");  
  
      search[keyValue[0]] = keyValue[1];  
    }  
  
    return search;  
  }  
  
  PersistentSearch.prototype.getListItem = function() {  
    if (!this.listItem) {  
      this.listItem = newNode("div");  
      this.listItem.className = "lk cs listItem";  
      this.listItem.appendChild(newText(this.label));  
      this.resultSizeItem = newNode("span");  
      this.listItem.appendChild(this.resultSizeItem);  
      this.getResultSize(false);  
      var _this = this;  
      this.listItem.addEventListener('click', function() {  
getObjectMethodClosure(_this, "execute")(); }, true);  
    }  
  
    return this.listItem;  
  }  
  
  PersistentSearch.prototype.getEditItem = function() {  
    if (!this.editItem) {  
      this.editItem = newNode("tr");  
      this.editItem.className = "editItem";  
  
      var labelCell = newNode("td");  
      labelCell.className = "labelCell";  
      var labelInput = newNode("input");  
      labelInput.value = this.label;  
      labelCell.appendChild(labelInput);  
      this.editItem.appendChild(labelCell);  
  
      var editCell = newNode("td");  
      editCell.className = "editCell";  
      var queryInput = newNode("input");  
      queryInput.value = this.getEditableQuery();
```

```
        editCell.appendChild(queryInput);
        this.editItem.appendChild(editCell);
    }

    return this.editItem;
}

PersistentSearch.prototype.execute = function() {
    var searchForm = getNode("s");
    searchForm.elements.namedItem('q').value = this.getRunnableQuery();
    top.js._MH_OnSearch(unsafeWindow, 0);
}

PersistentSearch.prototype.getRunnableQuery = function() {
    var query = this.query;

    var today = new Date();
    var yesterday = new Date(today.getTime() - ONE_DAY);
    var oneWeekAgo = new Date(today.getTime() - 7 * ONE_DAY);

    query = query.replace(/:today/g, ":" + getDateString(today));
    query = query.replace(/:yesterday/g, ":" + getDateString(yesterday));
    query = query.replace(/:oneweekago/g, ":" + getDateString(oneWeekAgo));

    return query;
}

PersistentSearch.prototype.getEditableQuery = function() {
    return this.query;
}

PersistentSearch.prototype.getResultSize = function(needsRefresh) {
    if (this.totalResults == -1 || this.unreadResults == -1) {
        needsRefresh = true;
    } else {
        this.updateResultSizeItem();
    }

    if (needsRefresh) {
        this.resultSizeItem.style.display = "none";
        this.runQuery(this.getRunnableQuery(),
            getObjectMethodClosure(this, "getUnreadResultSize"));
    }
}

PersistentSearch.prototype.runQuery = function(query, continuationFunction)
{
    var queryUrl = "http://mail.google.com/mail?search=query&q=" +
        escape(query) + "&view=tl";

    GM_xmlhttpRequest({method: 'GET', url: queryUrl,
```

```

onload: function(oResponseDetails) {
  var match = RESULT_SIZE_RE.exec(oResponseDetails.responseText);
  if (match) {
    var resultSize = match[3];
    continuationFunction(resultSize);
  } else {
    alert("Couldn't find result size in search query.");
  }
}

PersistentSearch.prototype.getUnreadResultSize = function(totalResults) {
  this.totalResults = totalResults;

  this.runQuery(this.getRunnableQuery() + " is:unread",
    getObjectMethodClosure(this, "updateResultSize"));
}

PersistentSearch.prototype.updateResultSize = function(unreadResults) {
  this.unreadResults = unreadResults;

  savePersistentSearches();

  this.updateResultSizeItem();
}

PersistentSearch.prototype.updateResultSizeItem = function() {
  if (this.resultSizeItem) {
    // Clear existing contents
    var child;

    this.resultSizeItem.style.display = "";

    while (child = this.resultSizeItem.firstChild) {
      this.resultSizeItem.removeChild(child);
    }

    // Update with new values
    this.resultSizeItem.appendChild(newText(" ("));
    var unread = newNode(this.unreadResults > 0 ? "b" : "span");
    unread.appendChild(newText(this.unreadResults));
    this.resultSizeItem.appendChild(unread);
    this.resultSizeItem.appendChild(newText("/" + this.totalResults + "));
  }
}

function initializeStyles() {
  var styleNode = newNode("style");

  document.body.appendChild(styleNode);

  var styleSheet = document.styleSheets[document.styleSheets.length - 1];

  for (var i=0; i < RULES.length; i++) {
    styleSheet.insertRule(RULES[i], 0);
  }
}

```

```

}
}

initializeStyles();
initializePersistentSearches();

```

### Running the Hack

After installing the user script (Tools → Install This User Script), log into your Gmail account at <http://mail.google.com>. You will see a yellow box in the sidebar, between Labels and Invites.

By default, the new box displays three searches. Click a search to execute it, and the results will display in the standard messages pane, as shown in Figure 7-11.



Figure 7-11. Results of saved search

Each saved search shows the number of unread and total messages that match it. These are cached; to update them, click on the refresh icon in the upper-right corner of the box.

You can also edit your saved searches by clicking the “Edit searches” link, as shown in [Figure 7-12](#).

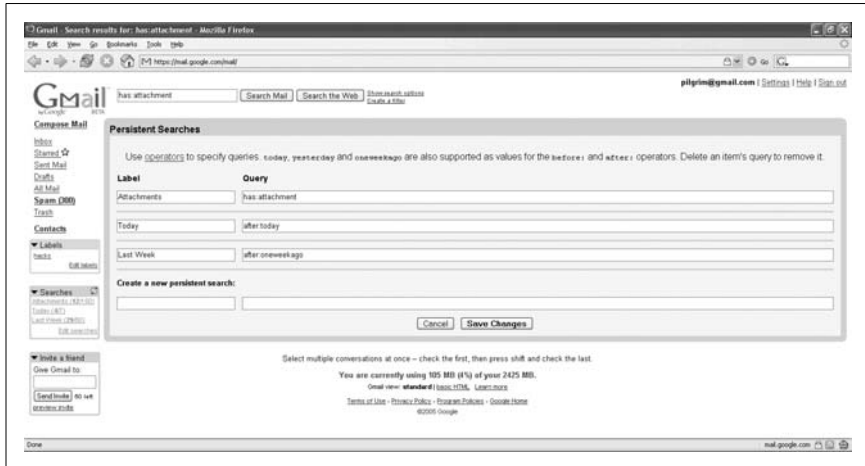


Figure 7-12. Editing saved searches

You can use standard Gmail search syntax in your saved searches, as well as a few custom operators such as `before:` and `after:`.

—Mihai Parparita