Trying To Not Use All Your Memory

Robert O'Callahan
Mozilla Corporation
Please be interactive.
Credit to Nick Nethercote and many others.
• Our problem space
• A glimpse into Firefox memory management
• What we haven't figured out yet
Once upon a time, Web browsers were simple.
The company pays me ten dollars for every bug I fix in my code, Ratbert.

I want you to do your little rat dance on my keyboard so I'll have lots of bugs to fix.

How am I doing?

Not so good. You just authored a web browser.
Things have changed.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>types_attribute_returns_a_DOMStringList</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>When_drag_data_store_s_mode_is_in_the_protected_mode_getData_returns_the_empty_string</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Forms</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>tgkt [tests relating to data list]</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>history</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>pablof history_state_is_supported</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>pablof history.state_has_an_initial_value_of_null</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>pablof history.state_is_updated_by_pushState</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Sandbox</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>The_iframe_element_s_sandbox_DOM_property Uses_the_DOMSettableTokenList_interface</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>SVG</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>The_SVGElement_supports_getIntersectionList_and_getEnclosureList_with_all_renderable_elements</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Property_inheritance_and_the_clipPath_element</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>The_filter_property_applied_to_container_and_indirectly_renderable_elements</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>CSS</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>@namespace_rule_must_precede_all_other_non-ignored_rule_sets_in_a_style_sheet</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>tomz enabled_on_a_fieldset_element</td>
</tr>
</tbody>
</table>
| 24 |   | http://samples.msdn.microsoft.com/ietestcenter/css3/namespaces/syntax-
New Legal Permanent Residents in the U.S. (per year) vs. U.S. Population vs. U.S. History

One Point Of View

- Resource loading
- HTML/CSS rendering
- Canvas drawing API
- DOM API
- Storage API
Host Objects

- DOM API objects implemented by browser in C++
- FFI/“DOM bindings” very important
- Memory management across language boundary very important
  - Especially cycles
Additional Constraints

- 100s of tabs
- KB to GB per tab
- Page load/unload churn
- 60FPS
- “The Web” is difficult to characterize and evolves rapidly
Commodity Software

- Users compare browser memory usage, share impressions, and switch browsers
- Reducing memory usage matters even if it has no impact on performance
- Must release memory ASAP when closing tabs while user is watching Task Manager
- Must be competitive even on extremely poorly designed Web sites
- Worst-case performance matters
Memory Management In Firefox

- JS heap: incremental mark and sweep collector
  - WIP: Moving generational
- C++ objects: reference counting with smart pointers
- Everything: cycle collector
  [Bacon+Rajan, ECOOP01]
"It [reference counting] ... is unused by mature high performance systems."

— An ISMM 2012 paper
Cycle Collection
Cycle Collection

Node marked purple/"suspect" when refcount decremented.

Live purple nodes are “roots” of potential cycles. CC does not require explicit knowledge of root set (win!)
Cycle Collection

Mark purple nodes and those reachable from purple as “gray”. Count number of incoming edges found for each node.
Cycle Collection

Traverse gray nodes breadth-first, starting with the former purple nodes:

- If all references found, then it's garbage; release it later.
- Otherwise it's live: preserve it and all gray nodes reachable from it.
Cycle Collector

✔ Works with C++ (albeit manual tracing)
✔ Edges and objects that can't be involved in cycles don't need tracing
✔ Only looks at potential garbage not already released by reference counting

● “Everything live” is a common steady state
● Can delay CC until a certain amount of potential garbage exists

✗ Not fully generational/incremental (yet)
Optimizing Cycle Collection

Skip purple node if we can quickly determine it is live
Optimizing Cycle Collection

Application-specific fast liveness test for big wins.

Generalize this!
Javascript Compartments

http://example.com

Wrapper

http://example.com/subframe

window

Wrapper list

Security, accounting, GC, CPG
Firefox had a reputation for memory usage.
Built better measurement tools.
Explicit Allocations

651.44 MB (100.0%) -- explicit
  361.86 MB (55.55%) -- js
  201.65 MB (30.95%) -- window-objects
    93.09 MB (14.29%) -- top(http://www.whatwg.org/specs/web-apps/current-w...)
    92.63 MB (14.22%) -- window(http://www.whatwg.org/specs/web-apps/current-w...)
      50.86 MB (07.81%) -- layout
        35.01 MB (05.37%) -- arenas
        15.74 MB (02.42%) -- pres-contexts
        0.12 MB (00.02%) -- style-sets
      41.65 MB (06.39%) -- dom [2]
    0.12 MB (00.02%) -- style-sheets
  0.45 MB (00.07%) -- (2 tiny)
  51.05 MB (07.84%) -- (44 tiny)
  16.39 MB (02.52%) -- top(http://dxr.lanedo.com/mozilla-central/content/...)
  15.68 MB (02.41%) -- top(https://tbpl.mozilla.org/?tree=Try, id=0107)/a...
  10.06 MB (01.54%) -- top(https://mail.google.com/mail/u/0/?ui=2&shva=1#...)
  7.80 MB (01.20%) -- top(https://dxr.lanedo.com/mozilla-central/xpcom/ba...
  7.59 MB (01.16%) -- top(https://bugzilla.mozilla.org/show_bug.cgi?id=5...
  7.47 MB (01.15%) -- window(https://bugzilla.mozilla.org/show_bug.cgi...)
  0.11 MB (00.02%) -- window(about:blank)
  25.56 MB (03.92%) -- images
  23.08 MB (03.54%) -- gfx
  19.93 MB (03.06%) -- storage
  10.14 MB (01.56%) -- network-memory-cache
  9.23 MB (01.42%) -- (7 tiny)
Found and fixed many bugs.
Bugs Found

- Actual leaks
- Bloated data structures
- Space allocated but never used
- Non-Firefox issues: leaky addons and sites
sqlite3_int64 *p;
nByte = ROUND8(nByte);
p = malloc( nByte+8 );
if( p ){
    p[0] = nByte;
    p++;
    p++;
}

nByte is normally an SQLite page size, a power of 2...
• Nick used instrumentation to find and fix many such issues
Unscientific Benchmark

Lifehacker, Feb 2012
Blocking Addon-related Leaks

Browser UI → Addon → twitter.com → facebook.com → google.com
Blocking Addon-related Leaks

Browser UI
Addon

twitter.com
facebook.com

google.com
Blocking Addon-related Leaks

Browser UI

Addon

twitter.com

facebook.com

goole.com
Lessons Learned

• Need measurement tools users can run
• Need good tools for Web developers and Firefox addon developers
• Still difficult to debug some bugs:

  “I ran Firefox for a week and leaked some memory”
Thoughts for the future:
How far can you push refcounting + cycle collection?
Interactive applications demand 60fps.

Not much time for GC pauses or VM page-in.

End of virtual memory?
Divergence between client and server workloads.
Without virtual memory, how should apps cooperate to optimize memory usage?

“OOM killing” is popular, but suboptimal. “ashmem” difficult to use.
Applications make isolated caching decisions based on little data and less principle.
Foolproof abstractions that Web developers can use to optimize memory usage across a pool of apps?
Valuable negative results: Solutions that should work but don't.
Questions?

Clownshoes: http://www.flickr.com/photos/29233640@N07/5131195458/

Pig: http://www.flickr.com/photos/22864665@N06/5082987037/