Big boost to DOM performance - WebKit's innerHTML is 240% faster

By Sam Dutton
Sam is a Developer Advocate

We're very happy to see that some common DOM operations have just skyrocketed in speed. The changes were at the WebKit level, boosting performance for both Safari (JavaScriptCore) and Chrome (V8).

Chrome Engineer Kentaro Hara made seven code optimisations within WebKit; below are the results, which show just how much faster JavaScript DOM access has become:

DOM performance boosts summary

- `div.innerHTML` and `div.outerHTML` **performance improved by 2.4x** (V8, JavaScriptCore)
- `div.innerText` and `div.outerText` performance in Chromium/Mac **by 4x** (V8/Mac)
- CSS property accesses improved **by 35%** (JavaScriptCore)
- `div.classList`, `div.dataset` and `div.attributes` perf improved by **up to 10.9x** (V8)
- `div.firstElementChild`, `lastElementChild`, `previousElementSibling` and `nextElementSibling` perf improved **by 7.1x** (V8)
- V8 DOM attributes access improved **by 4 ~ 5%** (V8)

Below, Kentaro Hara gives details on some of the patches he made. The links are to WebKit bugs with test cases, so you can try out the tests for yourself. The changes were made between WebKit r109829 and r111133: Chrome 17 does not include them; Chrome 19 does.

Improve performance of `div.innerHTML` and `div.outerHTML` by 2.4x (V8, JavaScriptCore)

Previous behavior in WebKit:

1. Create a string for each tag.

2. Append a created string to `Vector<string>`, parsing the DOM tree.
3. After the parsing, allocate a string whose size is the sum of all strings in the `Vector<string>`.
4. Concatenate all strings in `Vector<string>`, and return it as `innerHTML`.

New behavior in WebKit:

1. Allocate one string, say S.
2. Concatenate a string for each tag to S, incrementally parsing the DOM tree.
3. Return S as `innerHTML`.

In a nutshell, instead of creating a lot of strings and then concatenating them, the patch creates one string and then simply append strings incrementally.

Improve performance of `div.innerText` and `div.outerText` in Chromium/Mac by 4x (V8/Mac)

The patch just changed the initial buffer size for creating `innerText`. Changing the initial buffer size from 2^16 to 2^15 improved Chromium/Mac performance by 4x. This difference depends on the underlying malloc system.

Improve performance of CSS property accesses in JavaScriptCore by 35%

(Note: This is a change for Safari, not for Chrome.)

A CSS property string (e.g. `.fontWeight`, `.backgroundColor`) is converted to an integer ID in WebKit. This conversion is heavy. The patch caches the conversion results in a map (i.e. a property string => an integer ID), so that the conversion won't be conducted multiple times.

How do the tests work?

They measure the time of property accesses. In case of `innerHTML` (the performance test in bugs.webkit.org/show_bug.cgi?id=81214), the test just measures the time to run the following code:

```
for (var i = 0; i < 1000000; i++)
    document.body.innerHTML;
```

The performance test uses a large body copied from the HTML spec.

Similarly, the CSS property-accesses test measures the time of the following code:

```
var spanStyle = span.style;
for (var i = 0; i < 1000000; i++) {
    spanStyle.invalidFontWeight;
    spanStyle.invalidColor;
    spanStyle.invalidBackgroundColor;
    spanStyle.invalidDisplay;
}
```

The good news is that Kentaro Hara believes more performance improvements will be possible for other important DOM attributes and methods.

Bring it on!

Kudos to Haraken and the rest of the team.

Note: Demo removed
