Некоторые специфические замеры скорости яваскрипта

18 января 2010 г.

Аннотация

Описывается небольшое исследование производительности браузерного яваскрипта, которое потребовалось чтобы понять узкие места, удостовериться в паре предположений и сделать выбор между selectors api и xpath.

Целью исследования является выяснение средней скорости работы в достаточно современном браузере отдельных конструкций яваскрипта. Маловероятно, что результаты настоящего тестирования позволяют судить объективно о превосходстве скоростных возможностей одного браузера над другим.

1 Цели работы

При проектировании новой версии пеночки возникли типичные проблемы проектирования: стало непонятно, как поведут себя некоторые патерны, которые планировалось часто использовать. В частности для сокрытия информации в модулях использовалась следующая конструкция:

```
;(function () {
   ...module code here...
})();
```

Она конечно слишком простая для того, чтобы тормозить, однако практика показала, что с развитием скрипта количество возможностей в нем, а значит и количество модулей, имеет тенденцию переходить все

разумные пределы. Поэтому стало страшно и захотелось померять скорость.

Также в ходе безумных исследований интернетов с помощью гугла было установлено, что в браузерах на сегодняшний день существуют две примерно одинаковые технологии выборки нужного из DOM'а — CSS-селекторы (функции querySelector, querySelectorAll) и XPath (в основном функция document.evaluate и некоторые другие). Автоматически встала проблема выбора какой-либо одной технологии, в ходе решения которой данные о скорости выборки обоими методами были бы не лишни.

В-третьих обнаружилось, что родные браузерные события — довольно медленная штука, не предназначенная для того, чтобы ими кидались как попало, а кидаться как попало было надо: OOD помимо объектов, имеющих внутреннее, а значит скрытое состояние подразумевает еще и возможность обмена сообщениями. Поэтому нужен механизм обмена сообщениями, причем быстрый механизм. Чтобы окончательно удостовериться в тормознутости нативных событий браузеров нужно тоже замерить скорость их выполнения.

Наконец, несколько замеров производительности дадут некую общую температуру по палате: можно будет примерно понять, чего следует ожидать от яваскрипта.

2 Методика исследования

Для тестирования производительности использовалась экспериментальный прототип новой версии пеночки. Этим убивалось три зайца: (1) тестировалась производительность, (2) тестировалась производительностьи именно пеночки и (3) обкатывались её несущие конструкции. Исходник для тестирования приведен в приложении.

Все тестирование состоит из пяти тестов:

Namespace Тест конструкции предназначенной для сокрытия в модуле. В цикле такая конструкция создавалась миллион раз, время чего и замерялось.

Penochka msgs Собственная реализация событий (подробности см. в исходнике, модуль events). В хэше создавалось порядка 100 событий (чтобы браузеру было из чего выбирать), на какое-то одно вешалось

10 обработчиков и потом оно вызывалось 100000 раз. То есть опять таки миллион итераций.

Native msgs Браузерная реализация событий. Алгоритм такой же, как и в случае тестировани собственной реализации, только дополнительно 100 событий не создавалось — в браузере и своих хватает.

Xpath selection Выборка из ДОМа с помощью XPath. Из заглавной страницы сайта lenta.ru 1000 раз выбирались все гиперссылки (селектор `'//a'`).

CSS selection Выборка из ДОМа с помощью CSS. Из заглавной страницы lenta.ru также 1000 раз выбирались все гиперссылки (селектор `'a'`).

Производительность исследовалась на 7 браузерах: Опере 10.5, Опере 10.10, Хроме, Хромиуме, ФФ 3.0, ФФ 3.5, ФФ 3.6.

Измерения проводились на компьютере следующей конфигурации:

```
Название ОС:                    Microsoft Windows 7 Максимальная
Версия ОС:                      6.1.7600 Н/Д сборка 7600
Сборка ОС:                      Multiprocessor Free
Тип системы:                    x64-based PC
Процессор(ы):                   Число процессоров - 1.
                                [01]: AMD64 Family
Полный объем физической памяти: 1.791 МБ
```

В ходе измерений на том же компьютере играла музычка (ogg vorbis и mp3).

3 Результаты

Результаты сведены в следующую табличку. Преведено среднее время выполнения заданного теста заданным браузеров в миллисекундах.

Browser	**O 10.5**	**O 10.10**	**Chr**	**Chrm**	**FF3.0**	**FF3.5**	**FF3.6**
Namespace	725	1747	250	310	4000	2200	1950
Penochka msgs	808	2080	530	940	1900	1500	1300
Native msgs	E1	E1	3950	4300	9000	6700	5600
Xpath selection	1160	3400	1200	1280	7500	4100	3700
CSS selection	1380	3200	1150	1210	NS	4800	5100

Если в табличке число оканчивается на круглую цифру (нуль), то это означает, что в этих разрядах цифры варьировались от измерения к измерению, то есть по сути это отклонения из-за разного рода погрешностей.

Названия браузеров означают: **O 10.5** — Opera 10.5 (сборка 3172), **O 10.10** — Opera 10.10 (сборка 1893), **Chr** — Google Chrome (версия 3.0.195.38), **Chrm** — Chromium 4.0.302.0 (сборка 36495), **FF3.0** — Mozilla Firefox 3.0.13, **FF3.5** — Mozilla Firefox 3.5.7, **FF3.6** — Namoroka 3.6a1.

Если в ячейке таблицы стоит не число, а аббревиатура, то произошло следующее:

NS Нужный функционал не поддерживается.

E1 В ходе тестирования Опера планомерно съела всю физическую память и вылетела с ошибкой «Недостаточно памяти», даже не предложив отправить отчет об ошибке.

4 Анализ

Проведенное тестирование позволяет говорить о:

1. Скорость создания конструкции, скрывающей информацию достаточна для использования и не должна стать причиной падения производительности.

2. Собственная реализация событий всегда быстрее браузерного варианта в 4 раза. Также падение оперы в ходе тестирования говорит о том, что использовать эти события кроме как для подписки на всякие `'click'` или `'mousemove'` не стоит. Хотя с другой стороны такие плачевные результаты могут намекать на то, что методика измерений была неверная и кое-кто не умеет обращаться с браузерными событиями как следует.

3. Селекторы XPath и CSS имеют примерно одинаковую и достаточную для использования производительность. Вместе с тем CSS селекторы не поддерживаются Firefox'ом версии 3.0, что не добавляет им очков.

4. Хоть точность результатов и достаточна, скрипт тестирования содержит потенциал для получения более точных значений. Его улучшение является вопросом будующего.

Приложение. Исходные текст скрипта для тестирования

```
/** kernel - Основные функции

    Определение глобального объекта пеночки. Некоторые, удобные в
использовании функции, аналогичные по своей сущности библиотеке
jQuery. */
;(function () {
var window = this,
       undefined,
       pn = window.pn = function (el) {
          return new pn.fn.init(el)
       }

       pn.fn = pn.prototype = {
          init: function (el) {
     if (el && el.fn)
      return el

     return pn.fn.merge([el ? el : document])
  },
  merge: function (arr) {
     this.length = 0;
     if (toString.call(arr) === "[object Array]") {
      Array.prototype.push.apply(this, arr)
     } else {
      Array.prototype.push.apply(this, [arr])
     }
     return this
  }
       }

       pn.extend = pn.fn.extend = function (code) {
          for (method_name in code) {
     pn.fn[method_name] = code[method_name]
  }
```

```
      }
})();

/** events - Синхронные события

    Механизм подписки на системные события браузера и обмена
внутренними сообщениями */
;(function () {
var events = {}

 pn.on = pn.fn.on = function(evname, fun, issys, iscap) {
    if (issys != null) {
       var dispatcher = this[0] || document
       dispatcher.addEventListener(evname, fun, iscap)
    } else {
       try {
          events[evname].push(fun)
       } catch (err) {
  events[evname] = [fun]
       }
    }
 },
 pn.to = pn.fn.to = function (evname, cookie) {
    try {
       for(var i = 0; i < events[evname].length; i++)
          events[evname][i](cookie)
    } catch (err) {}
 }

})();

/** xquery - XPath query

    Выборки элементов из DOM посредством xpath */
;(function () {

      pn.extend({
init: function (selector) {
```

```
   if (selector && selector.fn)
      return selector

    if (typeof selector === "string")
       return pn(document).xpfind(selector)

   return pn.fn.merge(selector || document)
},
xpfind: function (what) {
   var xr = document.evaluate(what, this[0],
         null, XPathResult.ANY_TYPE, null)
   var e = xr.iterateNext()
   this.length = 0

   while (e) {
      Array.prototype.push.apply(this, [e])
      e = xr.iterateNext()
   }

   return this
},
      })

   pn.fn.find = pn.fn.xpfind


})();

/** cquery - CSS query

    Выборки элементов из DOM посредством селекторов CSS */
;(function () {

     var Sizzle = function (selector, context) {
  return context.querySelectorAll(selector)
  }
```

```
      pn.extend({
init: function (selector) {
   if (selector && selector.fn)
      return selector

    if (typeof selector === "string")
       return pn(document).cssfind(selector)

   return pn.fn.merge(selector || document)
},
cssfind: function (what) {
   this.length = 0
var cr = Sizzle(what, this[0])

for (var i = 0; i < cr.length; i++) {
     Array.prototype.push.apply(this, [cr[i]])
}

   return this
}
      })

   pn.fn.find = pn.fn.cssfind

})();

/** speedtest - Тесты скорости яваскрипта

    Тесты производительности специфических для пеночки конструкций */
;(function () {

       function time (fun) {
          var start = new Date().getTime()
      fun()
      return (new Date().getTime()) - start
       }

       var TEST_ITERATIONS = 1000000;
```

```
 window.namespace_create_test = function () {
    var a = 0
  var test = function () {
    for (var i = 0; i < TEST_ITERATIONS; i++) {
   (function () {
    a = a + 1
   })()
      }
    }

    var retval = time(test);
    return a == TEST_ITERATIONS ? retval : 'test failed';
 }

 window.message_pass_test = function () {
    var events = {}

  var AVG_EVENTS = 100;
  var AVG_HANDLERS = 10;

  var a = 0;

  for (var i = 0; i < AVG_EVENTS; i++) {
        pn.on('event'+ i, function () {})
  }

for (var i = 0; i < AVG_HANDLERS; i++) {
        pn.on('event10', function(e) { a = a + 1 })
  }

  function test () {
        for (var i = 0; i < TEST_ITERATIONS / AVG_HANDLERS; i++) {
  pn.to('event10', null)
        }
  }

 var retval = time(test);
```

```
       return a == TEST_ITERATIONS ? retval : 'test failed';
}

window.nativemessage_pass_test = function () {
      var AVG_HANDLERS = 10;

       var a = 0;

       for (var i = 0; i < AVG_HANDLERS; i++) {
             document.addEventListener('event10',
                  function(e) { a = a + 1 }, true)
       }

       function test () {
             for (var i = 0; i < TEST_ITERATIONS / AVG_HANDLERS; i++) {
         var evObj = document.createEvent('Events');
   evObj.initEvent('event10', false, false)
   document.dispatchEvent(evObj)
             }
       }

   var retval = time(test);
       return a == TEST_ITERATIONS ? retval : 'test failed';
   }

window.xpath_test = function () {
   var DIVIDER = 1000
var a = null

function test () {
         for (var i = 0; i < TEST_ITERATIONS / DIVIDER; i++) {
     a = pn(document).xpfind('//a')
         }
    }

var retval = time(test);
    return a ? retval : 'test failed';
}
```

```
window.selectors_test = function () {
   var DIVIDER = 1000
var a = null

function test () {
         for (var i = 0; i < TEST_ITERATIONS / DIVIDER; i++) {
     a = pn(document).cssfind('a')
         }
    }

var retval = time(test);
    return a ? retval : 'test failed';
}


})();
```