УДК 330.46

К.Ю. Островська

РОЗРОБКА РОЗШИРЕННЯ ПІДРАХУНКУ «ВТРАЧЕНОГО ЧАСУ» В INTERNET

Анотація. Розроблено розширення для Google Chrome для підрахунку «Втраченого часу» в Інтернет. Це розширення підраховує час, проведений на сайтах з можливістю визначення відвіданих сайтів за категоріями: корисний час, або втрачений час.

Ключові слова: втрачений час, Google Chrome, Internet, розширення.

Інтернет - це інструмент, який допомагає нам вчитися, спілкуватися, творити і т. д. Але освоювати нові технології або сервіси буває непросто, і це приводить багатьох в замішання. І початківцю, і досвідченому фахівцеві важливо знати, як працює той чи інший продукт і як домогтися від нього максимальної ефективності.

Розширення - це персоналізовані функції, які легко підключити до Google Chrome. Розширення дозволяють додавати в Google Chrome тільки потрібні функції, уникаючи тих, які вам не потрібні.

Розширення дозволяють покращувати сторінку шляхом додавання релевантних посилань і інформації.

Деякі розширення додають кнопки в область біля адресного рядка, дозволяючи отримувати повідомлення про певні події.

Деякі розширення виконують функції ярликів.

Розробка розширення для Google Chrome, створеного для підрахунку «втраченого часу» в Інтернеті. Тобто це розширення рахує час, проведений на сайтах з можливістю визначення відвіданих сайтів за категоріями: корисний час, або втрачений час.

Отже, почнемо зі створення розширення: зі створення папки самого розширення, в яку будемо складати всі створювані нами файли. Назва якого буде - «losttime». Далі, створимо файл manifest.json, який виглядає наступним чином:

```
manifest.json
 {
  "manifest_version": 2,
  "name": "Lost Time",
  "version": "1.0",
  "icons": {
    "128": ""
  },
  "content_scripts": [
    {
      "matches": [ "*://*/*" ],
      "js": [ "content.js" ]
    }
  ],
  "background": {
    "scripts": ["background.js"]
  },
  "permissions": [
    "http://losttime.su/*"
  ],
      "browser_action": {
        "default_title": "LostTime",
        "default_icon": "",
        "default_popup": "popup.html"
      }
}
```

Деякі з рядків мають бути інтуїтивно зрозумілі, але що обов'язково потрібно знати:

- Значення manifest_version має бути обов'язково «2»;

- У content_scripts пишемо, який скрипт буде запускатися на всіх сторінках окремо;

- У background пишемо загальний скрипт (фоновий скрипт), який запускається при запуску браузера;

- У permissions пишемо адресу сайту, з якого буде братися інформація.

JSON (англ. Java Script Object Notation) - текстовий формат обміну даними , заснований на JavaScript і зазвичай використовуваний саме з цією мовою. Як і багато інших текстові формати, JSON легко читається людьми.

Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA - 262 1999 года), формат вважається язиконезавісімим і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує готовий код для створення та обробки даних у форматі JSON.

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш підходящим для сериализации складних структур. Якщо говорити про веб -додатках, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP - інтерфейси).

Оскільки формат JSON є підмножиною синтаксису мови JavaScript, то він може бути швидко десеріалізован вбудованою функцією `eval ()`. Крім того, можлива вставка цілком працездатних JavaScript- функцій. У мові PHP, починаючи з версії 5.2.0, підтрим-

ка JSON включена в ядро у вигляді функцій `json_decode ()` і `json_encode ()`, які самі перетворять типи даних JSON у відповідні типи PHP і навпаки.

JSON - текст являє собою (в закодованому вигляді) одну з двох структур:

Набір пар ключ: значення. У різних мовах це реалізовано як об'єкт, запис , структура, словник, хеш -таблиця, список з ключем або асоціативний масив. Ключем може бути тільки рядок , значенням - будь-яка форма.

Впорядкований набір значень. У багатьох мовах це реалізовано як масив, вектор, список або послідовність.

Це універсальні структури даних: як правило, будь-який сучасний мова програмування підтримує їх у тій чи іншій формі. Вони лягли в основу JSON, так як він використовується для обміну даними між різними мовами програмування.

Як значення в JSON використовуються структури:

Об'єкт - це неврегульована безліч пар ключ: значення, укладену в фігурні дужки " {} ". Ключ описується рядком, між ним і значенням стоїть символ ":". Пари ключ- значення відокремлюються один від одного комами.

Масив (одновимірний) - це впорядкована безліч значень. Масив полягає у квадратні дужки " [] ". Значення розділяються комами.

Значення може бути рядком у подвійних лапках, числом, об'єктом, масивом, одним з літералів: true, false або null. Т.ч. структури можуть бути вкладені один в одного.

Рядок - це впорядкована безліч з нуля або більше символів юнікоду, укладену в подвійні лапки. Символи можуть бути вказані з використанням escape - послідовностей , що починаються із зворотного косою риси "\ ".

Рядок дуже схожа на однойменний тип даних в мовах C і Java. Число теж дуже схоже на C- або Java -число, за винятком того , що використовується тільки десятковий формат . Прогалини можуть бути вставлені між будь-якими двома синтаксичними елементами .

На рисунку 1 переставлене вікно, яке Ви можете бачити по кліку на іконку розширення - це сторінка: popup.html.

Рисунок 1 – Вікно статистики

Файл popup.html виглядає наступним чином:

```
<!doctype html>
<html>
  <head>
    <title>Потерянное время
LostTime</title>
        <script src="jquery.js"
type="text/javascript"></script>
<!-- Подключаю jquery -->
    <link href="css.css"
rel="stylesheet"
type="text/css"/><!-- Подключаю
стили-->
  </head>
  <body>
  <div id="options"><!-- меню -->
<a  href="/popup.html"><img
class='img' src="" Title =
"Короткая статистика за
сегодня"></a>
<a href="/options.html"><img
class='img'
src="images/options.png"
Title="Настройки программы"></a>
<a href="/stat.html"><img
class='img' src="images/stat.png"
Title="Подробная статистика о
посещаемости"></a>
</div>
<div id="dannie"></div> <!-- в
этот блок буду загружать данные,
которые будут показываться
пользователю-->
    <script
src="popup.js"></script><!--
скрипт, выполняющийся при нажатии
на иконку расширения-->
  </body>
</html>
```

Щоб було зрозуміліше, опис коду представлено на мові гіпертекстової розмітки HTML. Меню організуємо просто: на картинку необхідно поставити внутрішню посилання розширення.

Файл popup.js виглядає вельми просто:

```
var xhr = new XMLHttpRequest();
xhr.open("GET",
"http://losttime.su/?tmpl=login&t
oken="+localStorage['lostlogin'],
true); // тут происходит ГЕТ
запрос на указанную страницу
xhr.onreadystatechange =
function() {
  if (xhr.readyState == 4) //
если всё прошло хорошо,
выполняем, что в скобках
  {
    var dannie =
document.getElementById('dannie')
;
    dannie.innerHTML =
xhr.responseText; // добавляем в
блок с id=dannie  полученный код
  }
}
xhr.send();
```

 ISSN 1562-9945

Саме описана вище конструкція дозволяє витягти і вивести зміст з Вашого, а може і не з Вашого сайту.

Важливо знати:

- У файлі маніфесту обов'язково в полі permissions пишемо адресу сайту, з якого буде братися інформація.

- Файл popup.js пов'язаний з фоновим скриптом background.js, т.к. дані, занесені в локальне сховище на background.js, видно і на popup.js.

Перед тим, як розглянути файл фонового скрипта background.js, розглянемо файл скрипта, який запускається на кожній сторінці окремо - content.js.

```
content.js
function onBlur() { // окно
теряет фокус

chrome.runtime.sendMessage({site:
sait,time:localStorage[sait]});
// отправка сообщения на
background.js
    localStorage[sait] = '0';
}
    window.onblur = onBlur; //
если окно теряет фокус
    function sec() //выполняется
каждую секунду
    {

if(document.webkitVisibilityState
== 'visible')//если страница
активна
        {
           localStorage[sait] =
parseInt(localStorage[sait],10)
+1; // обновляем данные о сайте в
локальном хранилище
        }
    }
var sait=location.hostname; // на
каком сайте находится скрипт
localStorage[sait] = '0';
    setInterval(sec, 1000);//
запускать функцию каждую секунду
```

Найбільш цікавий момент з скрипта, повинен бути:

`chrome.runtime.sendMessage ({site: sait, time: localStorage [sait]});`

Тут відбувається відправлення повідомлення background скрипту, а саме дві змінні: site: sait - містить адресу сайту, на якому скрипт

`time: localStorage [sait]` - кількість часу, проведена на цьому скрипті.

Далі, розглянемо фоновий скрипт background.js, де і відбувається прийом даних, а точніше розглянемо саму функцію прийому даних.

```
background.js
chrome.runtime.onMessage.addListener(
  function(request, sender, sendResponse) {
```

```
var a = request.site; // данные о сайте
var b = request.time; // данные о проведенном времени
// тут делаем с этими данными что хотим.
  });
```

Якщо в скрипті background.js додати які-небудь дані в локальне сховище, то ці ж дані можна буде використовувати і в popup.js скрипті.

На сторінці налаштувань необхідно було організувати перетягування сайтів в різні колонки, рисунок 2.

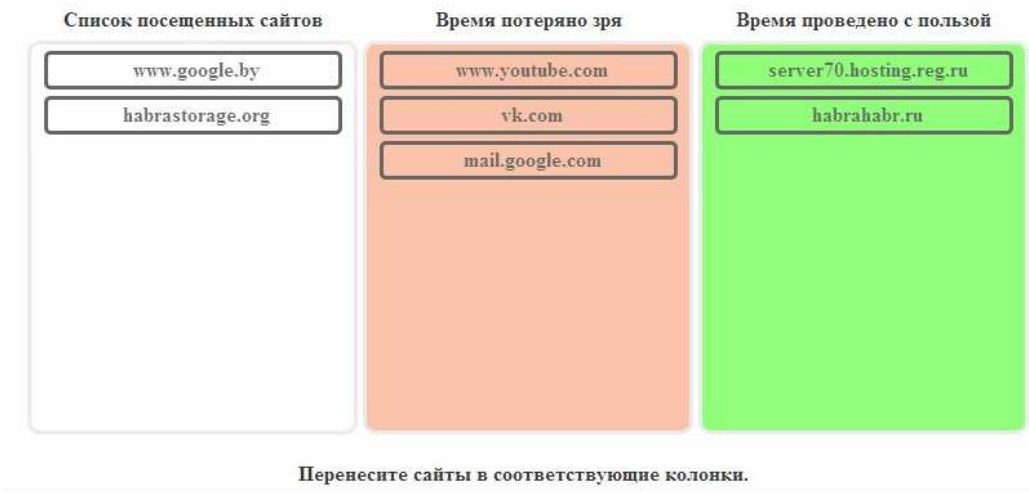

Рисунок 2 - Налаштування сайтів в різні колонки

Так як дані вставляються за допомогою InnerHtml, то дана можливість просто так не з'явиться. Ось, що довелося організувати:

```
      $('#dannie').on('mouseover'
, '.sait', function( ) {

$(this).css({'border':'3px   solid
#ffffff'});
      });
      $('#dannie').on('mouseout',
'.sait', function( ) {

$(this).css({'border':'3px   solid
black'});
      });
      $('#dannie').on('mousedown'
, '.sait', function( ) {

$(this).css({'border':'3px   solid
black'});
      });

      $('#dannie').on('mouseover'
, '.sait', function( ) {

$('.sait').draggable({
      helper:'clone'
          });
      });
```

Встановлює обробники подій на вибрані елементи сторінки. Має два варіанти використання:

```
on ( events, [ selector ], [ data ], handler ): jQueryv : 1.7
```

events - тип (и) оброблюваних подій. Наприклад "click", "resize" і т.д. (список всіх подій див. нижче). Якщо необхідно прив'язати обробник відразу на кілька типів подій, потрібно перерахувати їх через пропуск: "click resize..."

selector - селектор по якому будуть фільтруватися елементи, що лежать всередині вже знайдених. У підсумку, обробник буде спрацьовувати тільки в тому випадку, якщо подія «піднялося» від одного з відфільтрованих елементів.

data - дані, передані оброблювачу подій. У обробнику будуть доступні у змінній event.data.

handler - функція, яка буде встановлена в якості обробника. Замість функції, можна вказати значення false, це буде еквівалентно установці такої функції: function () { return false ;}.

.on (events - map, [selector], [data]): jQueryv : 1.7

за допомогою цього методу можна встановити на вибрані елементи відразу кілька різних обробників подій, кожен з яких буде реагувати на свій тип події.

events - map - об'єкт, в якому потрібно перерахувати типи оброблюваних подій і відповідні їм обробники. Здається в форматі:

{events - 1 : handler - 1, events - 2 : handler - 2,...},

де events - і handler - і відповідають параметрам events та handler в першому варіанті методу (описаному вище).

Тестування розширення.

Заходимо в Настройки - Інструменти - Розширення, тиснемо на «Завантажити розпаковане розширення».

ЛІТЕРАТУРА

1. Марко Беллиньясо Разработка Web-приложений в среде ASP.NET 2.0: задача - проект - решение = ASP.NET 2.0 Website Programming. - М.:»Диалектика», 2007. — С. 640.
2. Посилання в мережі Інтернет:
http://ru.wikipedia.org/wiki/Google_Chrome