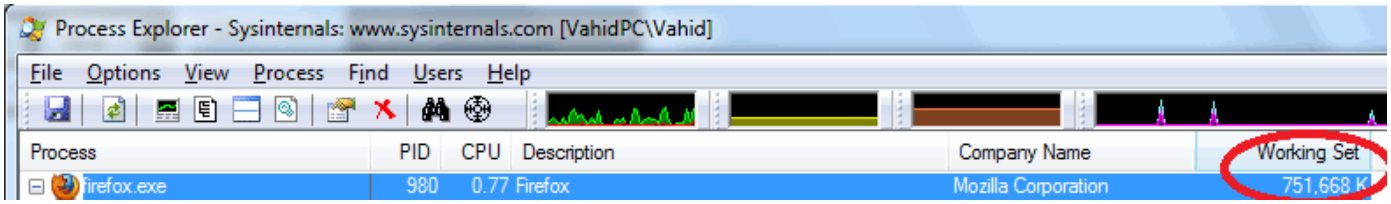
عنوان: **فایرفاکس 4 و غیرفعال کردن قابلیت تنظیم دستی اندازه جعبه‌های متنی آن**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۰۹ ۰۰:۰۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Firefox

احتمالا فایرفاکس 4 رو تازه نصب کردید:



یکی از موارد جالب توجه آن منهای مورد فوق، امکان تغییر سایز TextArea در آن به صورت "سر خود" می‌باشد (همانند مرورگر کروم) :

برای غیرفعال کردن این قابلیت باید css سایت یا عنصر مورد نظر را به صورت ذیل تغییر داد:

```
<style type="text/css">
textarea {
resize:none;
}
</style>
```

عنوان: **شرح یک مشکل امنیتی با فایرفاکس**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۱:۶
آدرس: www.dotnettips.info
برچسب‌ها: Firefox, Security

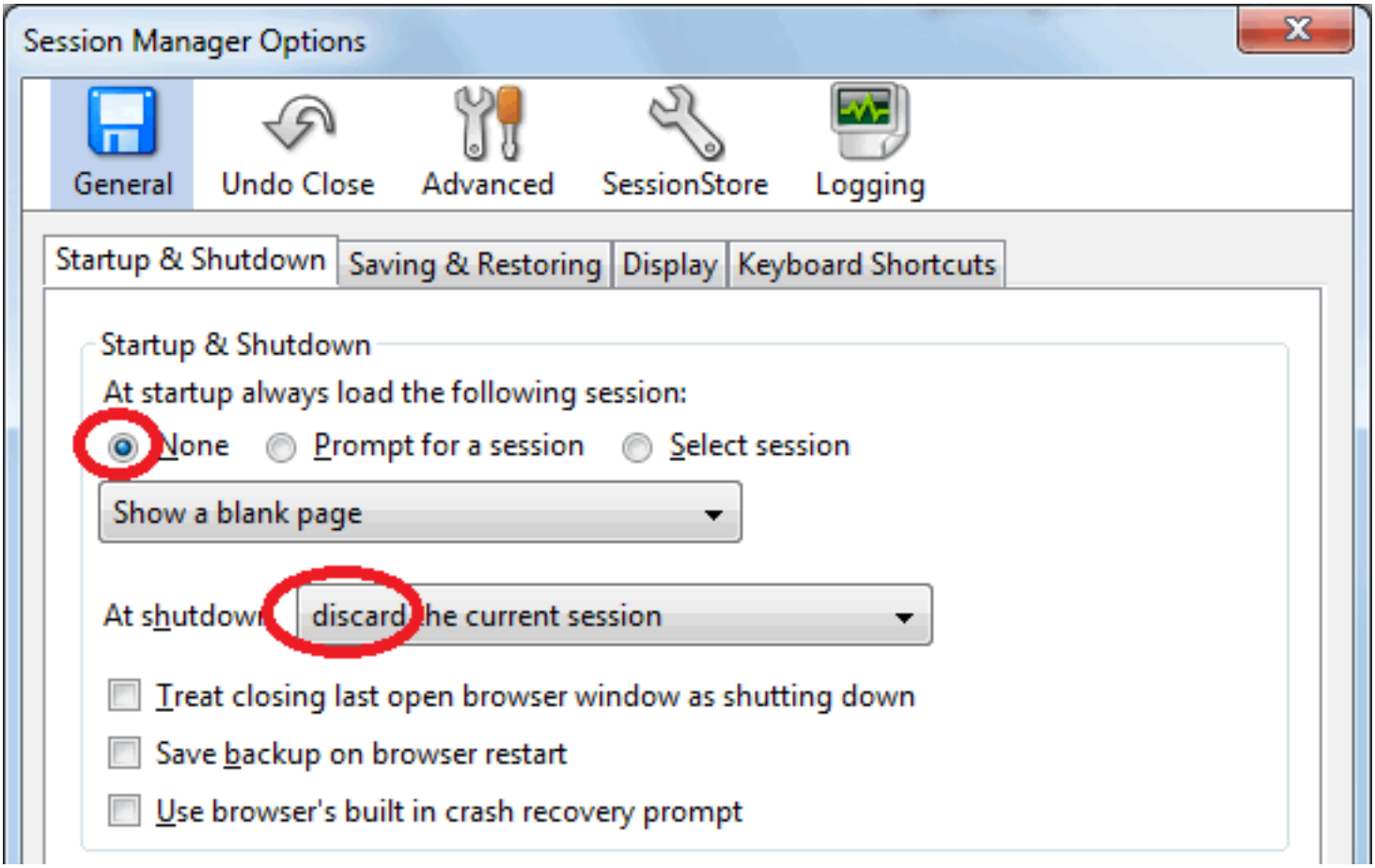
حدود دو ماه قبل دوبار از طریق میل‌باکس یاهو من به تمام contactهای تعریف شده در آن ایمیلی با محتوای زیر ارسال شده بود:

```
Hello,
you should definitely check this thing out http://www.newsl5.net/biz/?page=xyz
```

این ایمیل‌ها هم جعلی نبودند. یعنی واقعا از اکانت یاهوی من ارسال شده بودند و در قسمت sent وجود خارجی داشتند! فقط IP ارسال کننده آن (115.78.224.246) متعلق به کشور ویتنام بود (IP ارسال کننده را در هدر ایمیل ارسالی می‌توان مشاهده کرد).
این مساله باعث شد که من سیستم را چندین بار چک کنم؛ از لحاظ بحث ویروس تا اسپای‌ور و غیره. «هیچ» مشکلی مشاهده نشد.
مرحله بعد کمی در مورد یاهو جستجو کردم و مشخص شد که یاهو با session hijacking به شدت مشکل دارد. همچنین ابزار دیگری که می‌تواند به این session hijacking کمک کند خود «فایرفاکس» است. فایرفاکس حاوی گزینه‌ای است که سشن‌های قبلی شما را ذخیره می‌کند. زمانیکه مرورگر را بسته و پس از مدتی آن‌را باز می‌کنیم، یک راست و قشنگ همان سشن قبلی مثلا یاهو را بازیابی کرده و کار ادامه پیدا می‌کند.
کمی گشتم و این قابلیت رو به کل غیرفعال کردم. برای غیرفعال کردن آن «Disable Session Restore in Firefox» را در گوگل جستجو کنید.
و خلاصه آن به صورت زیر است:
در نوار آدرس فایرفاکس تایپ کنید about:config
در ادامه موارد زیر را یافته و غیرفعال کنید:

```
browser.sessionstore.resume_from_crash;false
browser.sessionstore.resume_session_once;false
browser.sessionstore.restore_pinned_tabs_on_demand;false
browser.sessionstore.restore_hidden_tabs;false
services.sync.prefs.sync.browser.sessionstore.restore_on_demand;false
```

راه ساده‌تر:
افزونه session manager را نصب کنید
در قسمت session manager options در برگه startup & shutdown آن کلا بحث ذخیره سازی سشن در حین بسته شدن مرورگر را غیرفعال کنید.



و به صورت خلاصه: تنظیمات پیش فرض فایرفاکس از لحاظ امنیتی مناسب نیستند.
ضمن اینکه ایمیل فوق رو من هفته‌ای یکی دو بار از تمام افرادی که می‌شناسم دریافت می‌کنم! به عبارتی خیلی‌ها گرفتار این مساله شده‌اند.
ذخیره سازی سشن‌ها به نظر کارها رو ساده می‌کنه. مرورگر رو باز می‌کنی همه چیز مثل قبل از بسته شدن آن است و ... همین یعنی مشکل امنیتی. خصوصا مراجعه به سایت‌ها و لینک‌هایی که از باگ‌های XSS سوء استفاده می‌کنند.

نظرات خوانندگان

نویسنده: میثم هوشمند
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۴:۱۱

خب یعنی فایر فاکس مشکل امنیتی دارد؛ و این مشکلی که برای شما به وجود آمده به دلیل ورود به سایتی بوده که حمله XSS صورت داده است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۶:۲۹

بله. این مشکلی که نام بردم خیلی دامنه دار است. حدود چندماهی است که مدام برای تمام آشناهای من ارسال شده و همه مشکل پیدا کردن.
علت اینکه امروز این مطلب رو نوشتم دریافت مجدد چندباره یک چنین ایمیلی از آشناها بود.
مشخصات آن هم این است که به تمام contactهای تعریف شده شما ارسال شده و در قسمت sent قابل مشاهده است.

نویسنده: احسان
تاریخ: ۱۳۹۱/۰۴/۰۳ ۲۰:۲۳

من ابتدا session manager رو استفاده کردم و تنظیمات رو طبق راهنما انجام دادم
ولی هنوز browser.sessionstore.resume_from_crash در حالت فعال بود
بنابراین به صورت دستی هم کار رو انجام دادم که خاطر جمع باشه

عنوان: **افزونه نویسی برای مرورگرها : فایرفاکس : قسمت اول**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۱/۰۵ ۲۲:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Firefox, Plugin, Extension, Addon

در دو مقاله پیشین ^ ، ^ به بررسی نوشتن افزونه در مرورگر کروم پرداختیم و اینبار قصد داریم همان پروژه را برای فایرفاکس پیاده کنیم. پس در مورد کدهای تکراری توضیحی داده نخواهد شد و برای فهم آن می‌توانید به دو مقاله قبلی رجوع کنید. همه‌ی ما فایرفاکس را به خوبی می‌شناسیم. اولین باری که این مرورگر آمد سرو صدای زیادی به پا کرد و بازار وسیعی از مرورگرها را که در چنگ IE بود، به دست آورد . این سر و صدا بیشتر به خاطر امنیت و کارآیی بالای این مرورگر، استفاده از آخرین فناوری‌های تحت وب و دوست داشتنی برای طراحان وب بود. همچنین یکی دیگر از مهمترین ویژگی‌های آن، امکان سفارشی سازی آن با افزونه‌ها extensions یا addon بود که این ویژگی در طول این سال‌ها تغییرات زیادی به خود دیده است. در مورد افزونه نویسی برای فایرفاکس در سطح نت مطالب زیادی وجود دارند که همین پیشرفت‌های اخیر در مورد افزونه‌ها باعث شده خیلی از این مطالب به روز نباشند. اگر در مقاله پیشین فکر می‌کنید که کروم چقدر در نوشتن افزونه جذابیت دارد و امکانات خوبی را در اختیار شما می‌گذارد، الان دیگر وقت آن است که نظر خودتان را عوض کنید و فایرفاکس را نه تنها یک سرو گردن بلکه بیشتر از این حرف‌ها بالاتر بدانید.
شرکت موزیالا برای قدرتمندی و راحتی کار طراحان یک sdk طراحی کرده است است و شما با استفاده از کدهای موجود در این sdk قادرید کارهای زیادی را انجام دهید. برای نصب این sdk باید پیش نیازهایی بر روی سیستم شما نصب باشد:
نصب پایتون 2.5 یا 2.6 یا 2.7 که فعلا در سایت آن، نسخه‌ی 2.7 در دسترس هست. توجه داشته باشید که هنوز برای نسخه‌ی 3 پایتون پشتیبانی صورت نگرفته است.

آخرین نسخه‌ی sdk را هم می‌توانید از این آدرس به صورت zip و یا از این آدرس به صورت tar دانلود کنید و در صورتیکه دوست دارید به سورس آن دسترسی داشته باشید یا اینکه از سورس‌های مشارکت شده یا غیر رسمی استفاده کنید، از این صفحه آن را دریافت کنید.

بعد از دانلود sdk به شاخه‌ی bin رفته و فایل activate.bat را اجرا کنید. موقعی که فایل activate اجرا شود، باید چنین چیزی دیده شود:

```
(C:\Users\aym\Downloads\addon-sdk-1.17) C:\Users\aym\Downloads\addon-sdk-1.17\bin>
```

برای سیستم‌های عامل Linux,FreeBSD,OS X دستورات زیر را وارد کنید:
اگر یک کاربر پوسته‌ی bash هستید کلمه زیر را در کنسول برای اجرای activate بزنید:

```
source bin/activate
```

اگر کاربر پوسته‌ی بش نیستید:

```
bash bin/activate
```

نهایتا باید کنسول به شکل زیر در آید یا شبیه آن:

```
(addon-sdk)~/mozilla/addon-sdk >
```

بعد از اینکه به کنسول آن وارد شدید، کلمه cfx را در آن تایپ کنید تا راهنمای دستورات و سوییچ‌های آن‌ها نمایش داده شوند. از این ابزار میتوان برای راه اندازی فایرفاکس و اجرای افزونه بر روی آن، پکیج کردن افزونه، دیدن مستندات و آزمون‌های واحد استفاده کرد.

آغاز به کار

برای شروع، فایل‌های زیادی باید ساخته شوند، ولی نگران نباشید cfx این کار را برای شما خواهد کرد. دستورات زیر را جهت

ساخت یک پروژه خالی اجرا کنید:

```
mkdir fxaddon
cd fxaddon
cfx init
```

یک پوشه را در مسیری که کنسول بالا اشاره میکرد، ساختم و وارد آن شدم و با دستور cfx init دستور ساخت یک پروژه‌ی خالی را دادم و باید بعد از این دستور، یک خروجی مشابه زیر نشان بدهد:

```
* lib directory created
* data directory created
* test directory created
* doc directory created
* README.md written
* package.json written
* test/test-main.js written
* lib/main.js written
* doc/main.md written
Your sample add-on is now ready for testing:
try "cfx test" and then "cfx run". Have fun!"
```

در این پوشه یک فایل به اسم package.json هم وجود دارد که اطلاعات زیر داخلش هست:

```
{
  "name": "fxaddon",
  "title": "fxaddon",
  "id": "jid1-QfyqpNby9lTlcQ",
  "description": "a basic add-on",
  "author": "",
  "license": "MPL 2.0",
  "version": "0.1"
}
```

این اطلاعات شامل نام و عنوان افزونه، توضیحی کوتاه در مورد آن، نویسنده‌ی افزونه، ورژن افزونه و ... است. این فایل دقیقا معادل manifest.json در کروم است. در افزونه نویسی‌های قدیم این فایل install.rdf نام داشت و بر پایه‌ی فرمت rdf بود. ولی در حال حاضر با تغییرات زیادی که افزونه نویسی در فایرفاکس کرده‌است، الان این فایل بر پایه یا فرمت json است. اطلاعات package را به شرح زیر تغییر می‌دهیم:

```
{
  "name": "dotnettips",
  "title": ".net Tips Updater",
  "id": "jid1-QfyqpNby9lTlcQ",
  "description": "This extension keeps you updated on current activities on dotnettips.info",
  "author": "yeganehaym@gmail.com",
  "license": "MPL 2.0",
  "version": "0.1"
}
```

رابطهای کاربری Action Button و Toggle Button فایل main.js را در دایرکتوری lib باز کنید:
موقعی که در کروم افزونه می‌نوشتیم امکانی به اسم browser action داشتیم که در اینجا با نام action button شناخته می‌شود. در اینجا باید کدها را require کرد، همان کاری در خیلی از زبان‌ها مثلا مثل سی برای صدا زدن سرآیندها می‌کنید. مثلا برای action button اینگونه است:

```
var button= require('sdk/ui/button/action');
```

نحوه‌ی استفاده هم بدین صورت است:

```
buttons.ActionButton({...});
```

که در بین {} خصوصیات دکمه‌ی مورد نظر نوشته می‌شود. ولی من بیشتر دوست دارم از شیء دیگری استفاده کنم. به همین جهت ما از یک مدل دیگر button که به اسم toggle button شناخته می‌شود، استفاده می‌کنیم. از آن جا که این button دارای دو حالت انتخاب (حالت فشرده شده) و غیر انتخاب (معمولی و آماده فشرده شدن توسط کلیک کاربر) است، بهترین انتخاب هست.

کد زیر یک toggle button را برای فایرفاکس می‌سازد که با کلیک بر روی آن، صفحه‌ی popup.htm به عنوان یک پنل روی آن رندر می‌شود:

```
var tgbutton = require('sdk/ui/button/toggle');
var panels = require("sdk/panel");
var self = require("sdk/self");

var button = tgbutton.ToggleButton({
  id: "updaterui",
  label: ".Net Updater",
  icon: {
    "16": "./icon-16.png",
    "32": "./icon-32.png",
    "64": "./icon-64.png"
  },
  onChange: handleChange
});

var panel = panels.Panel({
  contentURL: self.data.url("./popup.html"),
  onHide: handleHide
});

function handleChange(state) {
  if (state.checked) {
    panel.show({
      position: button
    });
  }
}

function handleHide() {
  button.state('window', {checked: false});
}
```

در سه خط اول، فایل‌هایی را که نیاز است Required شوند، می‌نویسیم و در یک متغیر ذخیره می‌کنیم. اگر در متغیر نریزیم مجبور هستیم همیشه هر کدی را به جای نوشتن عبارت زیر:

```
tgbutton.ToggleButton
```

به صورت زیر بنویسیم:

```
require('sdk/ui/button/toggle').ToggleButton
```

که اصلا کار جالبی نیست. اگر مسیرهای نوشته شده را از مبدا فایل zip که اکسترکت کرده‌اید، در دایرکتوری sdk در شاخه lib بررسی کنید، با دیگر موجودیت‌های sdk آشنا خواهید شد.

در خط بعدی به تعریف یک شیء از نوع toggle button به اسم button میپردازیم و خصوصیاتی که به این دکمه داده ایم، مانند یک کد شناسایی، یک برچسب که به عنوان tooltip نمایش داده خواهد شد و آیکن‌هایی در اندازه‌های مختلف که در هرجایی کاربر آن دکمه را قرار داد، در اندازه‌ی مناسب باشد و نهایتا به تعریف یک رویداد می‌پردازیم. تابع handlechange زمانی صدا زده می‌شود که در وضعیت دکمه‌ی ایجاد شده تغییری حاصل شود. در خط بعدی شیء panel را به صورت global میسازیم. شیء self دسترسی ما را به اجزا یا فایل‌های افزونه خودمان فراهم می‌کند که در اینجا دسترسی ما به فایل html در شاخه‌ی data میسر شده است و مقدار مورد نظر را در contentURL قرار می‌دهد. نهایتا هم برای رویداد onhide تابعی را در نظر می‌گیریم تا موقعی که پنجره بسته شد بتوانیم وضعیت toggle button را به حالت قبلی بازگردانیم و حالت فشرده نباشد. چرا که این دکمه تنها با کلیک ماوس به حالت فشرده و حالت معمولی سوییچ میکند. پس اگر کاربر با کلیک بر روی صفحه‌ی مرورگر پنجره را ببندد، دکمه در همان وضعیت فشرده باقی می‌ماند.

همانطور که گفتیم تابع handlechnage موقعی رخ میدهد که در وضعیت دکمه، تغییری رخ دهد و نمیدانیم که این وضعیت فشرده شدن دکمه هست یا از حالت فشرده خارج شده است. پس با استفاده از ویژگی checked بررسی میکنم که آیا دکمه‌ای فشرده شده یا خیر؛ اگر برابر true بود یعنی کاربر روی دکمه، کلیک کرده و دکمه به حالت فشرده رفته، پس ما هم پنل را به آن نشان می‌دهیم و خصوصیات دلخواهی را برای مشخص کردن وضعیت پنل نمایشی به آن پاس می‌کنیم. خصوصیت یا پارامترهای زیادی را می‌توان در حین ساخت پنل برای آن ارسال کرد. با استفاده از خصوصیت position محل نمایش پنجره را مشخص می‌کنیم. در صورتی که ذکر نشود پنجره در وسط مرورگر ظاهر خواهد شد.

تابع onhide زمانی رخ میدهد که به هر دلیلی پنجره بسته شده باشد که در بالا یک نمونه‌ی آن را عرض کردیم. ولی اتفاقی که می‌افتد، وضعیت تابع را با متد state تغییر می‌دهیم و خصوصیت checked آن را false می‌کنیم. بجای پارامتر اولی، دو گزینه را میتوان نوشت؛ یکی window و دیگری tab است. اگر شما گزینه tab را جایگزین کنید، اگر در یک تب دکمه به حالت فشرده برود و به تب دیگر بروید و باعث بسته شدن پنجره بشوید، دکمه تنها در تبی که فعال است به حالت قبلی باز می‌گردد و تب اولی همچنان حالت خود را حفظ خواهد کرد پس می‌نویسیم window تا این عمل در کل پنجره اعمال شود.

Context Menus

برای ساخت منوی کانتکست از کد زیر استفاده می‌کنیم:

```
var contextMenu = require("sdk/context-menu");

var home = contextMenu.Item({
  label: "صفحه اصلی",
  data: "http://www.dotnettips.info/"
});
var postsarchive = contextMenu.Item({
  label: "مطالب سایت",
  data: "http://www.dotnettips.info/postsarchive"
});

var menuItem = contextMenu.Menu({
  label: "Open .Net Tips",
  context: contextMenu.PageContext(),
   items: [home, postsarchive],
  image: self.data.url("icon-16.png"),
  contentScript: 'self.on("click", function (node, data) {' +
                 '  window.location.href = data;' +
                 '});'
});
```

این منو هم مثل کروم دو زیر منو دارد که یکی برای باز کردن صفحه‌ی اصلی و دیگری برای باز کردن صفحه‌ی مطالب است. هر کدام یک برچسب برای نمایش متن دارند و یکی هم دیتا که برای نگهداری آدرس است. در خط بعدی منوی پدر یا والد ساخته می‌شود که با خصوصیت items، زیر منوهایش را اضافه می‌کنیم و با خصوصیت image، تصویری را در پوشه‌ی دیتا به آن معرفی می‌کنیم که اندازه‌ی آن 16 پیکسل است و دومی هم خصوصیت context است که مشخص می‌کند این گزینه در چه مواردی بر روی context menu نمایش داده شود. الان روی همه چیزی نمایش داده می‌شود. اگر گزینه، SelectionContext باشد، موقعی که متنی انتخاب شده باشد، نمایش می‌یابد. اگر SelectorContext باشد، خود شما مشخص می‌کنید بر روی چه مواردی نمایش یابد؛ مثلا عکس یا تگ p یا هر چیز دیگری، کد زیر باعث می‌شود فقط روی عکس نمایش یابد:

```
SelectorContext("img")
```

کد زیر هم روی عکس و هم روی لینکی که href داشته باشد:

```
SelectorContext("img,a[href]")
```

موارد دیگری هم وجود دارند که میتوانید مطالب بیشتری را در مورد آن‌ها در اینجا مطالعه کنید. آخرین خصوصیت باقی مانده، content script است که می‌توانید با استفاده از جاوااسکریپت برای آن کد بنویسید. موقعی که برای آن رویداد کلیک رخ داد، مشخص شود تابعی را صدا میزند با دو آرگومان؛ گره ای که انتخاب شده و داده‌ای که به همراه دارد که آدرس سایت است و آن را در نوار آدرس درج می‌کند.

آن منوهایی که با متد item ایجاد شده‌اند منوهایی هستند که با کلیک کاربر اجرا می‌شوند؛ ولی والدی که با متد menu ایجاد شده است، برای منویی است که زیر منو دارد و خودش لزومی به اجرای کد ندارد. پس اگر منویی میسازید که زیرمنو ندارد و خودش قرار است کاری را انجام دهد، به صورت همان item بنویسید که پایین‌تر نمونه‌ی آن را خواهید دید.

الان مشکلی که ایجاد می‌شود این است که موقعی که سایت را باز می‌کند، در همان تبی رخ می‌دهد که فعال است و اگر کاربر بر روی صفحه‌ی خاصی باشد، آن صفحه به سمت سایت مقصد رفته و سایت فعلی از دست میرود. روش صحیح‌تر اینست که تبی جدید بار شود و آدرس مقصد در آن نمایش یابد. پس باید از روشی استفاده کنیم که رویداد کلیک توسط کد خود افزونه مدیریت شود، تا با استفاده از شیء tab، یک تب جدید با آدرسی جدید ایجاد کنیم. پس کد را با کمی تغییر می‌نویسیم:

```
var tabs = require("sdk/tabs");
var menuItem = contextMenu.Menu({
  label: "Open .Net Tips",
  context: contextMenu.PageContext(),
   items: [home, postsarchive],
  image: self.data.url("icon-16.png"),
  contentScript: 'self.on("click", function (node, data) {' +
                 '  self.postMessage(data);' +
                 '});',
 onMessage: function (data) {
     tabs.open(data);
  }
});
```

با استفاده از postmessage، هر پارامتری را که بخواهیم ارسال می‌کنیم و بعد با استفاده از رویداد onMessage، داده‌ها را خوانده و کد خود را روی آن‌ها اجرا می‌کنیم.

بگذارید کد زیر را هم جهت سرچ مطالب بر روی سایت پیاده کنیم:

```
var Url="http://www.dotnettips.info/search?term=";
var searchMenu = contextMenu.Item({
  label: "search for",
  context: [contextMenu.PredicateContext(checkText),contextMenu.SelectionContext()],
    image: self.data.url("icon-16.png"),
  contentScript: 'self.on("click", function () {' +
  '  var text = window.getSelection().toString();' +
                 '  if (text.length > 20)' +
                 '   text = text.substr(0, 20);' +
                 '  self.postMessage(text);'+
                '})',
onMessage: function (data) {
```

```
    tabs.open(Url+data);
  }

});

function checkText(data) {

       if(data.selectionText === null)
           return false;

       console.log('selectionText: ' + data.selectionText);

       //handle showing or hiding of menu items based on the text content.
       menuItemToggle(data.selectionText);

       return true;
};

function menuItemToggle(text){
var searchText="جست و جو برای ";
    searchMenu.label=searchText+text;

};
```

در ساخت این منو، ما از ContextSelection استفاده کرده‌ایم. بدین معنی که موقعی که چیزی روی صفحه انتخاب شد، این منو ظاهر شود و گزینه‌ی دیگری که در کنارش هست، گزینه contextMenu.PredicateContext وظیفه دارد تابعی که به عنوان آرگومان به آن دادیم را موقعی که منو کانتکست ایجاد شد، صدا بزند و اینگونه میتوانیم بر حسب اطلاعات کانتکست، منوی خود را ویرایش کنیم. مثلا من دوست دارم موقعی که متنی انتخاب می‌شود و راست کلیک می‌کنم گزینه‌ی "جست و جو برای..." نمایش داده شود و به جای ... کلمه‌ی انتخاب شده نمایش یابد. به شکل زیر دقت کنید. این چیزی است که ما قرار است ایجاد کنیم:

در کل موقع ایجاد منو تابع checkText اجرا شده و متن انتخابی را خوانده به عنوان یک آرگومان برای تابع menuItemToggle ارسال می‌کند و به رشته "جست و جو برای" می‌چسباند. در خود پارامترهای آیتم اصلی، گزینه content scrip، با استفاده از جاوااسکریپت، متن انتخاب شده را دریافت کرده و با استفاده از متد postmessage برای تابع onMessage ارسال کرده و با ساخت یک تب و چسباندن عبارت به آدرس جست و جو سایت، کاربر را به صفحه مورد نظر هدایت کرده و عمل جست و جو در سایت انجام می‌گیرد.

در قسمت آینده موارد بیشتری را در مورد افزونه نویسی در فایرفاکس بررسی خواهیم کرد و افزونه را تکمیل خواهیم کرد

نظرات خوانندگان

نویسنده: مهدی رو
تاریخ: ۱۳۹۴/۰۳/۲۸ ۲۰:۴۶

با سلام؛ لطفا بگید چه طور قدم به قدم این کد هایی رو که مینویسیم اجرا کنیم؟ مرسی

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۲۸ ۲۲:۱۷

همینطور که در بالا گفتیم:
بعد از اینکه به کنسول آن وارد شدید، کلمه cfx را در آن تایپ کنید تا راهنمای دستورات و سوییچ‌های آن‌ها نمایش داده شوند. از این ابزار میتوان برای راه اندازی فایرفاکس و اجرای افزونه بر روی آن، پکیج کردن افزونه، دیدن مستندات و آزمون‌های واحد استفاده کرد.

از دستور زیر در مسیر دایرکتوری افزونه برای اجرا و تست افزونه استفاده می‌کنیم:

```
cfx run
```

انتهای مطلب قسمت دوم برای اجرا و بسته بندی افزونه این مورد رو توضیح دادیم.

عنوان: **افزونه نویسی برای مرورگرها : قسمت دوم : فایرفاکس**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۱/۱۵ ۱۰:۲۵
آدرس: www.dotnettips.info
گروه‌ها: Firefox, Extension, Addon, mozilla

در مقاله پیشین ، افزونه نویسی برای فایرفاکس را آغاز و مسائل مربوط به رابطهای کاربری را بررسی کردیم. در این قسمت که قسمت پایانی افزونه نویسی برای فایرفاکس است، به مباحث پردازشی و دیگر خصوصیت‌ها می‌پردازیم.
اولین موردی که باید برای برنامه‌ی ما در نظر گرفت، ذخیره و بازیابی مقادیر است که باید روی پنجره‌ی popup.html اعمال گردد و همچنین مقداردهی مقادیر پیش فرض برنامه بعد از نصب افزونه اعمال شود. برای ذخیره‌ی مقادیر، طبق نوشته موجود در راهنمای موزیلا، از روش زیر بهره برده و می‌توان مقادیر زیر را به راحتی در آن‌ها ذخیره کرد:

```
var ss = require("sdk/simple-storage");
ss.storage.myArray = [1, 1, 2, 3, 5, 8, 13];
ss.storage.myBoolean = true;
ss.storage.myNull = null;
ss.storage.myNumber = 3.1337;
ss.storage.myObject = { a: "foo", b: { c: true }, d: null };
ss.storage.myString = "O frabjous day!";
```

برای خواندن موارد ذخیره شده هم که مشخصا نوشتن اسم property کفایت می‌کند و برای حذف مقادیر نیز به راحتی از عبارت delete در جلوی پراپرتی استفاده کنید:

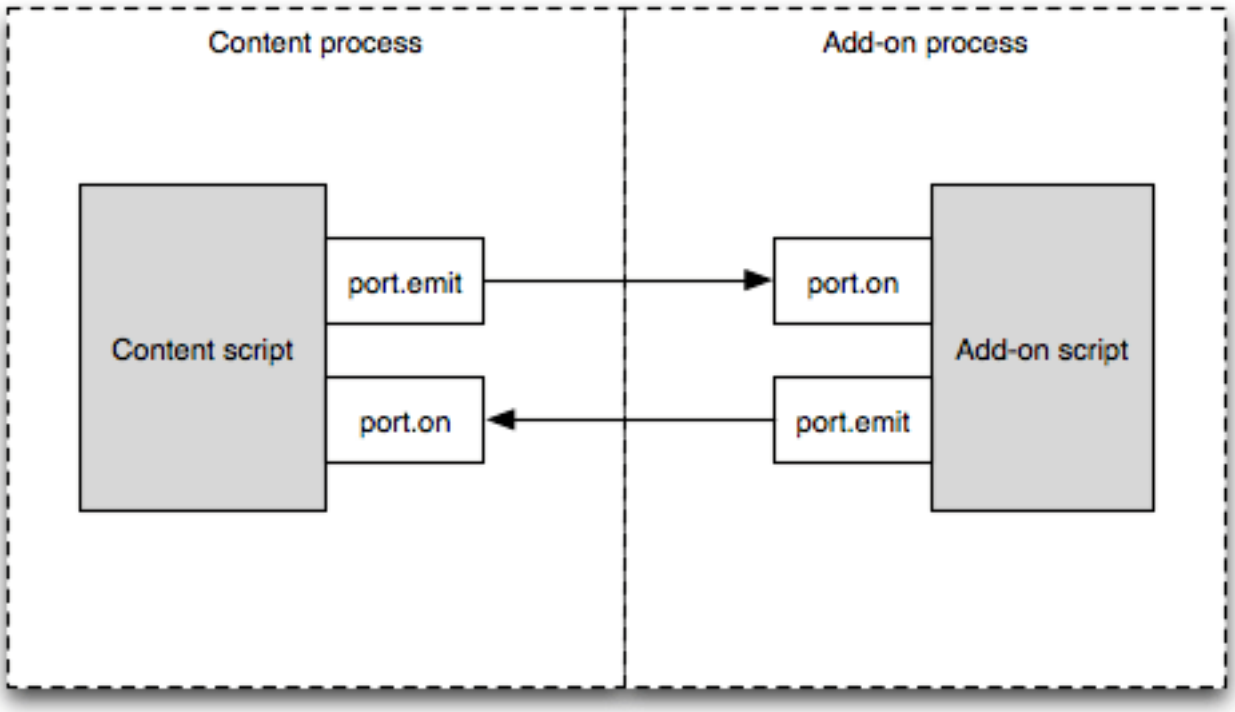
```
delete ss.storage.value;
```

برای ذخیره مقادیر پیش فرض اولین، کاری که می‌کنیم اسم متغیرها را چک می‌کنیم. اگر مخالف null بود، یعنی قبلا ست شده‌اند؛ ولی اگر null شد، عمل ذخیره سازی اولیه را انجام می‌دهیم:

```
if (!ss.storage.Variables)
 {
 ss.storage.Variables=[];
 ss.storage.Variables.push(true);
 ss.storage.Variables.push(false);
 ss.storage.Variables.push(false);
 ss.storage.Variables.push(false);
 }

if (!ss.storage.interval)
ss.storage.interval=1;

 if (!ss.storage.DateVariables)
 {
 var now=String(new Date());
 ss.storage.DateVariables=[];
 ss.storage.DateVariables.push(now);
 ss.storage.DateVariables.push(now);
 ss.storage.DateVariables.push(now);
 ss.storage.DateVariables.push(now);
 }
```

برای ذخیره مقادیر popup.html، به طور مستقیم نمی‌توانیم از کدهای بالا در جاوااسکریپت استفاده کنیم. مجبور هستیم که یک پل ارتباطی بین فایل main.js و فایل جاوااسکریپت داشته باشیم. در مقاله پیشین در مورد postmessage که ارتباطی از/به محتوا یا فایل جاوااسکریپت به/از main.js برقرار می‌کرد، صحبت کردیم و در این قسمت راه حل بهتری را مورد استفاده قرار می‌دهیم.
برای ایجاد چنین ارتباطی، آن هم به صورت دو طرفه از port استفاده می‌کنیم. دستور پورت در اکثر اشیایی که ایجاد می‌کنید وجود دارد ولی باز هم همیشه قبل از استفاده، از مستندات موزیلا حتما استفاده کنید تا مطمئن شوید دسترسی به شیء پورت در همه‌ی اشیا وجود دارد. ولی به صورت کلی تا آنجایی که من دیدم، در همه‌ی اشیا قرار دارد. از کد port.emit برای ارسال مقادیر به سمت فایل اسکریپت یا حتی بالعکس مورد استفاده قرار می‌گیرد و port.on هم یک شنونده برای آن است. شکل زیر به خوبی این

مبحث را نشان می‌دهد.



addon process در شکل بالا همان فایل main.js هست که کد اصلی addon داخل آن است و content process نیز محتوای اسکریپت است و حالا میتواند با استفاده از خاصیت contentscrip که به صورت رشته ای اعمال شده باشد یا اینکه با استفاده از خاصیت contentscriptfile، در یک یا چند فایل js استفاده نماید:

```
contentScriptFile: self.data.url("jquery.min.js")
contentScriptFile: [self.data.url("jquery.min.js"),self.data.url("const.js"),self.data.url("popup.js")]
```

از شیء port به صورت عملی استفاده می‌کنیم. کد main.js را به صورت زیر تغییر دادیم:

```
function handleChange(state) {
  if (state.checked) {
    panel.show({
      position: button
    });

 var v1=[],v2;
 if (ss.storage.Variables)
v1=ss.storage.Variables;

if (ss.storage.interval)
v2=ss.storage.interval;

panel.port.emit("vars",v1,v2);
  }
}
panel.port.on("vars", function (vars,interval) {
  ss.storage.Variables=vars;
  ss.storage.interval=interval;
});
```

در شماره پیشین گفتیم که رویداد handlechange وظیفه نمایش پنل را دارد، ولی الان به غیر آن چند سطر، کد دیگری را هم اضافه کردیم تا موقعی که پنل باز می‌شود، تنظیمات قبلی را که ذخیره کرده‌ایم روی صفحه نمایش داده شوند. با استفاده از شیء

port.emit محتواهای دریافت شده را به سمت فایل اسکریپت ارسال می‌کنیم تا تنظیمات ذخیره شده را برای نمایش اعمال کند. پارامتر اول رشته vars نام پیام رسان شما خواهد بود و در فایل مقصد هم تنها به پیامی با این نام گوش داده خواهد شد. این خصوصیت زمانی سودمندی خود را نشان می‌دهد که بخواهید در زمینه‌های مختلف، از چندین پیام رسان به سمت یک مقصد استفاده کنید. شیء panel.port.on هم برای گوش دادن به متغیرهایی است که از آن سمت برای ما ارسال می‌شود و از آن برای ذخیره‌ی مواردی استفاده میگردد که کاربر از آن سمت برای ما ارسال می‌کند. پس ما در این مرحله، یک ارتباطه کاملا دو طرفه داریم.

کد فایل popup.js که به صورت تگ script در popup.html معرفی شده است:

```
$(document).ready(function () {
addon.port.on("vars",  function(vars,interval) {
 if (vars)
{
 $("#chkarticles").attr("checked", vars[0]);
$("#chkarticlescomments").attr("checked", vars[1]);
$("#chkshares").attr("checked", vars[2]);
$("#chksharescomments").attr("checked", vars[3]);
}

$("#interval").val(interval);
});

    $("#btnsave").click(function() {

          var Vposts = $("#chkarticles").is(':checked');
          var VpostsComments = $("#chkarticlescomments").is(':checked');
          var  Vshares = $("#chkshares").is(':checked');
          var VsharesComments = $("#chksharescomments").is(':checked');
 var Vinterval = $("#interval").val() ;
 var Variables=[];
 Variables[0]=Vposts;
 Variables[1]=VpostsComments;
 Variables[2]=Vshares;
 Variables[3]=VsharesComments;
 interval=Vinterval;

 addon.port.emit("vars", Variables,Vinterval);
$("#messageboard").text( Messages.SettingsSaved);

    });
});
```

در همان ابتدای امر با استفاده از addon.port.on منتظر یک پیام رسان، به اسم vars می‌شود تا اطلاعات آن را دریافت کند که در اینجا بلافاصله بعد از نمایش پنل، اطلاعات برای آن ارسال شده و در صفحه، جایگذاری می‌کند. در قسمت رویداد کلیک دکمه ذخیره هم با استفاده از addon.port.emit اطلاعاتی را که کاربر به روز کرده است، به یک پیام رسان میدهیم تا برای آن سمت نیز ارسال کند تا در آن سمت، تنظیمات جدید ذخیره و جایگزین شوند.

نکته بسیار مهم: در کد بالا ما فایل جاوااسکریت را از طریق فایل popup.html معرفی کردیم، نه از طریق خصوصیت contentscriptfile. این نکته را همیشه به خاطر داشته باشید. فایل‌های js خود را تنها در دو حالت استفاده کنید:

از طریق دادن رشته به خصوصیت contentScript و استفاده از self به جای addon

معرفی فایل js داخل خود فایل html با تگ script که به درد اسکریپت‌های با کد زیاد می‌خورد.

اگر فایل شما شامل استفاده از کلمه‌ی کلیدی addon نمی‌شود، می‌توانید فایل js خود را از طریق contentScriptFile هم اعمال کنید. **فایل popup.html**

```
<script src="jquery.min.js"></script> <!-- Including jQuery -->
<script type="text/javascript" src="const.js"></script>
<script type="text/javascript" src="popup.js"></script>
```

خواندن فید RSS سایت

خواندن فید سایت توسط فایل Rssreader.js انجام می‌شود که تمام اسکریپت‌های مورد نیاز برای اجرای آن، توسط background.htm صدا زده شده است:

```
<script type="text/javascript" src="const.js"></script>
<script type="text/javascript" src="jquery.min.js"></script>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript" src="rssreader.js"></script>
```

تنها کاری که باید انجام دهیم اجرای این فایل به عنوان یک فرآیند پس زمینه است. در کروم ما عادت داشتیم برای این کار در فایل manifest.json از خصوصیت background استفاده کنیم، ولی از آنجا که خود فایل main.js یک فایل اسکریپتی است که در پس زمینه اجرا می‌شود، طبق منابع موجود در نت چنین چیزی وجود ندارد و این فرآیند را به خود فایل main.js مربوط می‌دانستند. ولی من با استفاده از page worker چنین خصوصیتی را پیاده سازی کردم. page worker وظیفه دارد تا یک آدرس یا فایلی را در یک تب پنهان و در پشت صحنه اجرا کرده و به شما اجازه‌ی استفاده از DOM آن بدهد. نحوه‌ی دسترسی به فایل background.htm توسط page worker به صورت زیر تعریف می‌شود:

```
pageWorker = require("sdk/page-worker");
  page= pageWorker.Page({
  contentScriptWhen: "ready",
  contentURL: self.data.url("./background.htm")
});
page.port.emit("vars",ss.storage.Variables,ss.storage.DateVariables,ss.storage.interval);
```

در فایل بالا شیء pageworker ساخته شد و درخواست یک پیج نهان را برای فایل background.htm در دایرکتوری data می‌کند. استفاده از گزینه‌ی contentScriptWhen برای دسترسی به شیء addon در فایل‌های جاوااسکریپتی که استفاده می‌کنید ضروری است. در صورتی که حذف شود و نوشته نشود با خطای *addon is not defined* روبرو خواهید شد، چرا که هنوز این شیء شناسایی نشده است. در خط نهایی هم برای آن سمت یک پیام ارسال شده که حاوی مقادیر ذخیره شده می‌باشد.

فایل RSSReader.js

در اینجا هم مانند مطلبی که برای کروم گذاشتیم، خواندن فید، در یک دوره‌ی زمانی اتفاق می‌افتد. در کروم ما از chrome.alarm استفاده می‌کردیم، ولی در فایرفاکس از همان تایمرهای جاوااسکریپتی بهره میبریم. کد زیر را به فایلی به اسم rssreader.js اضافه می‌کنیم:

```
var variables=[];
var datevariables=[];
var period_time=60000;
var timer;
google.load("feeds", "1");

$(document).ready(function () {
 addon.port.on("vars",  function(vars,datevars,interval) {
 if (vars)
{
Variables=vars;
}
if (datevars)
{
datevariables=datevars;
}
if(interval)
period_time=interval*60000;
alarmManager();
});
});

function alarmManager()
{
timer = setInterval(Run,period_time);
}

function Run() {
if(Variables[0]){RssReader(Links.postUrl,0, Messages.PostsUpdated);}
if(Variables[1]){RssReader(Links.posts_commentsUrl,1,Messages.CommentsUpdated); }
if(Variables[2]){RssReader(Links.sharesUrl,2,Messages.SharesUpdated);}
if(Variables[3]){RssReader(Links.shares_CommentsUrl,3,Messages.SharesCommentsUpdated);}
}

function RssReader(URL,index,Message) {
```

```
            var feed = new google.feeds.Feed(URL);
            feed.setResultFormat(google.feeds.Feed.XML_FORMAT);
                    feed.load(function (result) {
if(result!=null)
{
var strRssUpdate = result.xmlDocument.firstChild.firstChild.childNodes[5].textContent;
var RssUpdate=new Date(strRssUpdate);
var lastupdate=new Date(datevariables[index]);
if(RssUpdate>lastupdate)
{
datevariables[index]=strRssUpdate;
addon.port.emit("notification",datevariables,Message);
}

}
                        });
        }
```

در خطوط بالا متغیرها تعریف و توابع گوگل بارگزاری می‌شوند. سپس توسط addon.port یک شنونده ایجاد شده، تا بتواند مقادیر ذخیره شده را بازیابی کند. این مقادیر شامل موارد زیر است:

چه بخش‌هایی از سایت باید بررسی شوند.

آخرین تاریخ تغییر هر کدام که در زمان نصب افزونه، تاریخ نصب افزونه می‌شود و با اولین به روز رسانی، تاریخ جدیدی جای آن را می‌گیرد.

دوره‌ی سیکل زمانی یا همان interval بر اساس دقیقه

پس از اینکه شنونده مقادیر را دریافت کرد، تابع alarmManager اجرا شده و یک تایمر ایجاد می‌کند. بر خلاف کروم که برای این کار api تدارک دیده بود، اینجا شما باید از تایمرهای خود جاوااسکریپت مانند SetTimeout یا SetInterval استفاده کنید. موقع دریافت interval یا period_time ما آن را در 60000 ضرب کردیم تا دقیقه تبدیل به میلی ثانیه شود؛ چرا که تایمر، زمان را بر حسب میلی ثانیه دریافت می‌کند. وظیفه تایمر این هست که در هر دوره‌ی زمانی تابع Run را اجرا کند.

Run

این تابع بررسی می‌کند کاربر درخواست بررسی چه قسمت هایی از سایت را دارد و به ازای هر کدام، اطلاعات آن را از طریق پارامترها به تابع rssreader داده تا هر قسمت جداگانه بررسی شود. این اطلاعات به ترتیب: لینک فید مورد نظر، اندیس آخرین تاریخ به روزرسانی آن قسمت، پیامی که باید در وقت به روزرسانی به کار نمایش داده شود.

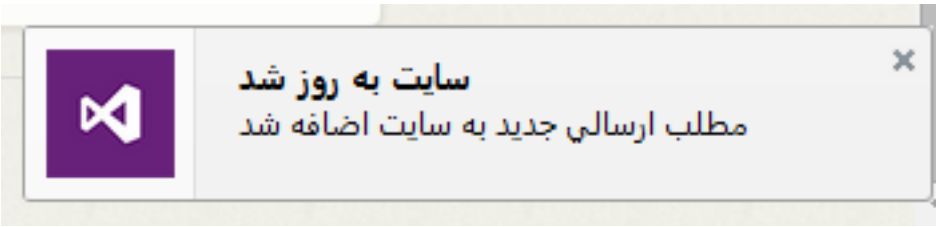
RSSReader

این تابع را قبلا در این مقاله توضیح دادیم. تنها تغییری که کرده است، بدنه‌ی شرط بررسی تاریخ است که در صورت موفقیت، تاریخ جدید، جایگزین تاریخ قبلی شده و یک پیام به فایل main.js ارسال می‌کند تا از آن درخواست ذخیره‌ی تاریخی جدید و هچنین ایجاد یک notification برای آگاه سازی کاربر کند. پس باز به فایل main.js رفته و شنونده آن را تعریف می‌کنیم:

```
page.port.on("notification",function(lastupdate,Message)
{
ss.storage.DateVariables=lastupdate;
Make_a_Notification(Message);
})
function Make_a_Notification(Message)
{
var notifications = require("sdk/notifications");
notifications.notify({
  title: "سایت به روز شد",
  text: Message,
  iconURL:self.data.url("./icon-64.png"),
  data:"http://www.dotnettips.info",
  onClick: function (data) {
tabs.open(data);
  }
});
}
```

شنونده مورد نظر دو پارامتر تاریخ آخرین به روزرسانی را دریافت کرده و آن را جایگزین قبلی می‌کند و پیام را به تابع Make_a_Notification پاس میکند. پارامترهای ساخت نوتیفیکیشن به ترتیب شامل عنوان، متن پیام، آیکن و نهایتا data است. دیتا شامل آدرس سایت است. زمانیکه کاربر روی نوتیفیکیشن کلیک می‌کند، استفاده شده و یک تب جدید را با آدرس سایت باز

می‌کنیم. به این ترتیب افزونه‌ی ما تکمیل می‌شود. برای اجرا و تست افزونه بر روی مرورگر فایرفاکس از دستور cfx run استفاده کنید.



البته این نکته قابل ذکر است که اگر کاربر طلاعات پنل را به روزرسانی کند، تا وقتی که مرورگر بسته نشده و دوباره باز نشود تغییری نمی‌کند؛ چرا که ما تنها در ابتدای امر مقادیر ذخیره شده را به RSSReader فرستاده و اگر کاربر آن‌ها را به روز کند، ارسال پیام دیگری توسط page worker صورت نمی‌گیرد. پس کد موجود در main.js را به صورت زیر ویرایش می‌کنیم:

```
pageWorker = require("sdk/page-worker");
  page= pageWorker.Page({
  contentScriptWhen: "ready",
  contentURL: self.data.url("./background.htm")
});

function SendData()
{
page.port.emit("vars",ss.storage.Variables,ss.storage.DateVariables,ss.storage.interval);
}
SendData();
panel.port.on("vars", function (vars,interval) {
  ss.storage.Variables=vars;
  ss.storage.interval=interval;
  SendData();
});
```

در کد بالا ما خطی که به سمت rssreader.js پیام ارسال می‌کند را داخل یک تابع به اسم SendDate قرار دادیم و بعد از تشکیل page worker آن را صدا زدیم و کد آن دقیقا مانند قبل است؛ با این تفاوت که اینبار این تابع را در جای دیگری هم صدا میزنیم و آن زمانی است که برای پنل، پیام مقادیر جدید ارسال می‌شود که در آن پس از ذخیره موارد جدید تابع SendData را صدا می‌زنیم. پس موقع به روزرسانی هم مقادیر ارسال خواهند شد. مقادیر جدید به سمت rssreader.js رفته و تشکیل یک تایمر جدید را می‌دهند و البته چون قبلا تایمر ایجاد شده است، پس باید چند خطی را هم به فایل rssreader.js اضافه کنیم تا تایمر قبلی را نابود کرده و تایمر جدیدی را ایجاد کند:

```
var timer;

function alarmManager()
{
timer = setInterval(Run,period_time);
}

 addon.port.on("vars",  function(vars,datevars,interval) {
 if (vars)
{
Variables=vars;

}
if (datevars)
{
datevariables=datevars;
}
if(interval)
period_time=interval*60000;

if(timer!=null)
{
clearInterval(timer);
```

```
}

alarmManager();
});
```

در خط بالا متغیری به اسم timer ایجاد شده است که کد timer را در خود ذخیره می‌کند. پس موقع دریافت مقادیر بررسی میکنیم که اگر مقدار timer مخالف نال بود تایمر قبلی را با clearInterval از بین برده و تایمر جدیدی ایجاد کند. پس مشکل تایمری که از قبل موجود است نیز حل می‌گردد.

افزونه‌ی ما تکمیل شد. اجازه بدهید قبل از بستن بحث چندتا از موارد مهم موجود در sdk را نام ببریم:

Page Mod page mod موقعی که کاربر آدرسی را مطابق با الگویی (pattern) که ما دادیم، باز کند یک اسکریپت را اجرا خواهد کرد:

```
var pageMod = require("sdk/page-mod");

pageMod.PageMod({
  include: "*.mozilla.org",
  contentScript: 'window.alert("Page matches ruleset");'
});
```

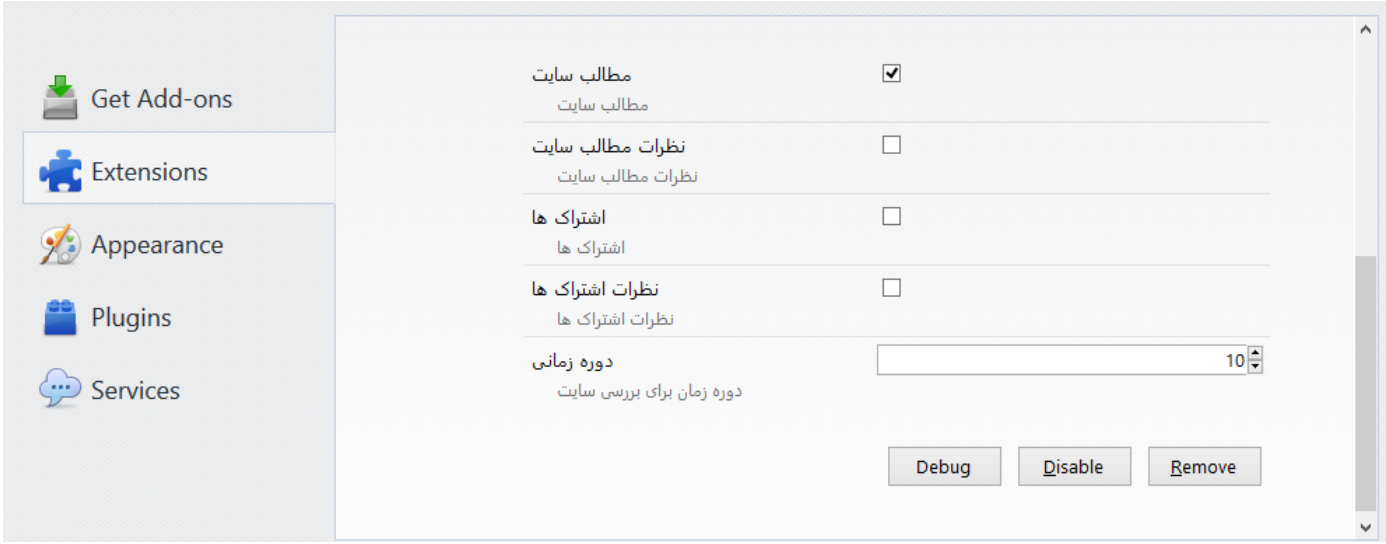
```
var data = require("sdk/self").data;
var pageMod = require("sdk/page-mod");

pageMod.PageMod({
  include: "*.mozilla.org",
  contentScriptFile: [data.url("jquery-1.7.min.js"),
                      data.url("my-script.js")]
});
```

پنل تنظیمات

موقعی که شما افزونه‌ای را در فایرفاکس اضافه می‌کنید، در پنلی که مدیریت افزونه‌ها قرار دارد می‌توانید در تنظیمات هر افزونه، تغییری ایجاد کنید. برای ساخت چنین صفحه‌ای از خصوصیت preferences در فایل package.json کمک می‌گیریم که مقادیر به صورت آرایه ای داخل آن قرار می‌گیرند. مثال زیر پنج کنترل را به بخش تنظیمات افزونه اضافه می‌کند که چهار کنترل اول چک باکس Checkbox هستند؛ چرا که خصوصیت type آنها به bool ست شده است و شامل یک نام و عنوان یا برچسب label و یک توضیح کوتاه است و مقدار پیش فرض آن با خصوصیت value مشخص شده است. آخرین کنترل هم یک کادر عددی است؛ چرا که خاصیت type آن با integer مقداردهی شده و مقدار پیش فرض آن 10 می‌باشد.

```
"preferences": [{
    "description": "مطالب سایت",
    "type": "bool",
    "name": "post",
    "value": true,
    "title": "مطالب سایت"
},
{
    "description": "نظرات مطالب سایت",
    "type": "bool",
    "name": "postcomments",
    "value": false,
    "title": "نظرات مطالب سایت"
},
{
    "description": "اشتراک ها",
    "type": "bool",
    "name": "shares",
    "value": false,
    "title": "اشتراک ها"
},
{
    "description": "نظرات اشتراک ها",
    "type": "bool",
    "name": "sharescomments",
    "value": false,
    "title": "نظرات اشتراک ها"
},
    {
        "description": "دوره زمان برای بررسی سایت",
        "name": "interval",
        "type": "integer",
```

```
        "value": 10,
        "title": "دوره زمانی"
    }]
```



از آنجا که مقادیر بالا تنها مقادیر پیش فرض خودمان هست و اگر کاربر آن‌ها را تغییر دهد، در این صفحه هم باید اطلاعات تصحیح شوند، برای همین از کد زیر برای دسترسی به پنل تنظیمات و کنترل‌های موجود آن استفاده می‌کنیم. همانطور که می‌بینید کد مورد نظر را در یک تابع به نام Perf_Default_Value قرار دادیم و آن را در بدو اجرا صدا زدیم. پس کاربر اگر به پنل تنظیمات رجوع کند، می‌تواند تغییراتی را که قبلا داده است، ببیند. بنابراین اگر الان تغییری را ایجاد کند، تا باز شدن مجدد مرورگر چیزی نمایش داده نمیشود. برای همین دقیقا مانند تابع SendData این تابع را هم در کد شنود پنل panel اضافه میکنیم؛ تا اگر کاربر اطلاعات را از طریق روش قبلی تغییر داد، اطلاعات هم اینک به روز شوند.

```
function Perf_Default_Value()
{
var preferences = require("sdk/simple-prefs").prefs;

preferences.post = ss.storage.Variables[0];
preferences.postcomments = ss.storage.Variables[1];
preferences.shares = ss.storage.Variables[2];
preferences.sharescomments = ss.storage.Variables[3];
preferences["myinterval"] =parseInt(ss.storage.interval);

}
Perf_Default_Value();

panel.port.on("vars", function (vars,interval) {
  ss.storage.Variables=vars;
  ss.storage.interval=interval;
  SendData();
  Perf_Default_Value();
});
```

البته کاربر فقط برای دیدن اطلاعات بالا به این صفحه‌ی تنظیمات نمی‌آید؛ بلکه بیشتر برای تغییر آن‌ها می‌آید. پس باید به تغییر مقدار کنترل‌ها گوش فرا دهیم. برای گوش دادن به تغییر تنظیمات، برای موقعی که کاربر قسمتی از تنظیمات را ذخیره کرد، از کدهای زیر بهره می‌بریم:

```
perf=require("sdk/simple-prefs");
var preferences = perf.prefs;
function onPrefChange(prefName) {

  switch(prefName)
  {
```

```
    case "post":
ss.storage.Variables[0]=preferences[prefName];
break;
case "postcomments":
ss.storage.Variables[1]=preferences[prefName];
break;
case "shares":
ss.storage.Variables[2]=preferences[prefName];
break;
case "sharescomments":
ss.storage.Variables[3]=preferences[prefName];
break;
case "myinterval":
ss.storage.interval=preferences[prefName];
break;
  }
}
//perf.on("post", onPrefChange);
//perf.on("postcomments", onPrefChange);

perf.on("", onPrefChange);
```

متد on دو پارامتر دارد: اولی، نام کنترل مورد نظر که با خصوصیت name تعریف کردیم و دومی هم تابع callback آن می‌باشد و در صورتی که پارامتر اول با "" مقداردهی شود، هر تغییری که در هر کنترلی رخ بدهد، تابع callback صدا زده می‌شود. از آنجا که نام کنترل‌ها به صورت string برگشت داده می‌شوند، برای دسترسی به مقادیر موجود در تنظیمات از همان روش داخل [""] بهره می‌گیریم. مقادیر را گرفته و داخل storage ذخیره می‌کنیم.

اشکال زدایی Debug

یکی از روش‌های اشکال زدایی، استفاده از console.log هست که میتونید برای بازبینی مقادیر و وضعیت‌ها، از آن استفاده کنید که نتیجه‌ی آن داخل کنسول نمایش داده می‌شود و اگر هم دربرنامه خطایی رخ دهد، داخل کنسول به شما نمایش خواهد داد.

سورس کار

نظرات خوانندگان

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۳/۱۱/۱۵ ۱۲:۳

از مطالب جذاب و کامل شما بسیار سپاسگذارم. چند نکته هست اگر امکان دارد آنها را نیز پوشش دهید
مثلا ارتباط این افزونه‌ها با بانکهای اطلاعاتی (مانند : localStorage مرورگر یا منابع داده دیگر مثل MySQL یا SQL Server و ...) و نحوه ذخیره سازی داده ها.
مجددا متشکرم.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۱/۱۵ ۱۵:۴

در مورد ذخیره سازی لوکال مرورگر که در بالا همان اول مقاله توضیح دادم و در کروم هم که گفتیم با کد زیر اینکارو انجام میدیم:

```
chrome.storage.local.set
chrome.storage.sync.set
```

این نکته را هم خاطرنشان کنم که در فایرفاکس ذخیره مقادیر تا حجم حدودی 5 مگابایت میسر است در مورد اتصال به دیتابیس sqlite میتونید از این لینک کمک بگیرید که به موارد دیگه هم لینک شده و اگر دقت کنید می‌بینید که میتوانید از کدهای c++ هم استفاده کنید و همینطور اینجا هم که یک نفر پرسش کرده و یکی هم پاسخش را داده.
در مورد بقیه اتصالات به بانک هایی چون sql server و ... هم میتوانید از طریق apiها یا وب سرویس‌ها عمل کنید که نیاز به یک فایل jquery برای اتصال به آن‌ها دارید یا فریمورک‌های جاوااسکریپتی که در این زمینه مهیا شده است.
این مقاله هم ممکنه براتون جالب باشه