برمجة إضافات جوجل كروم

متصفح جوجل كروم أصبح من أفضل المتصفحات على الإطلاق لما يوفره من أدوات و إضافات تسهل عملية التصفح و تجعل من تطوير الويب أمرا ممتعا بكل المقاييس.

لا شك في أنك جربت إضافات متصفح جوجل كروم من قبل،و مهما كانت هته الإضافة،فلا بد أنك تساءلت كيف تمت عملية برمجة و تطوير هته الإضافات،هل هذه العملية صعبة؟ما هي لغات البرمجة التي علي تعلمها حتى استطيع برمجة إضافات جوجل كروم الخاصة بك؟

أبشر،إن كنت مطور مواقع ويب فلن تحتاج لتعلم أية لغة برمجة أخرى،فإضافات متصفح جوجل كروم تتم برمجتها عن طريق لغات الويب الثلاثة: Java Script و CSS و HTML

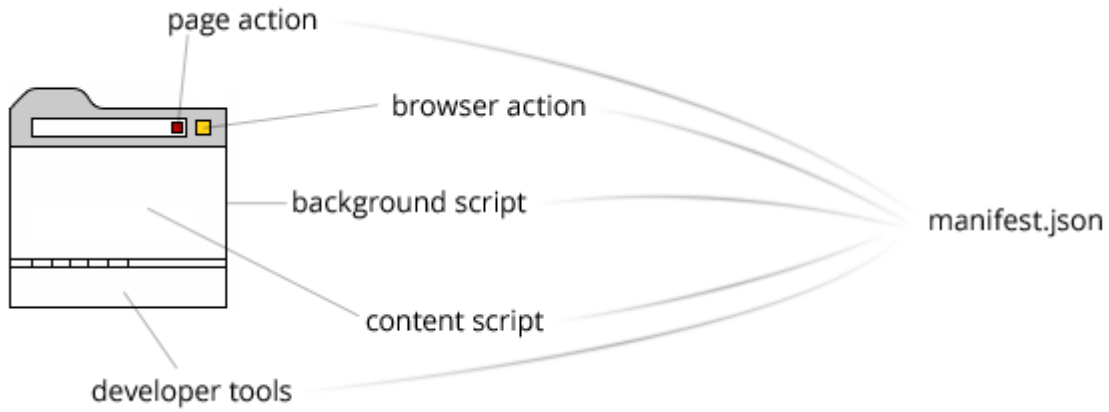
لا غير،و لكن عليك اتقان هته اللغات بشكل جيد جدا.

قبل أن نبدأ،دعونا نتعرف على بنية إضافات جوجل كروم أولا ثم ننتقل بعدها إلى كيف تتم البرمجة.عليك أن تقوم الآن بتفعيل Developer Mode من صفحة الإضافات:افتح المتصفح و اكتب في العنوان:

chrome://extensions

و لكن هذا بعد أن ننهي عملية البرمجة حتى لا نضطر إلى رفعها إلى متجر إضافات جوجل ثم نختار Load unpacked extension كروم في كل مرة نقوم بالتعديل على الإضافة.

الآن لنتعرف على بنية إضافات جوجل كروم:دعونا نرى الصورة الموالية



تتكون إضافات جوجل كروم من جزئين رئيسيين:الجزء البرمجي و واجهة المستخدم،و سنبدأ بالجزء البرمجي،و الذي بدوره ينقسم إلى ثلاثة أجزاء رئيسية:

و هو ملف أساسي لأي إضافة،يحوي هذا الملف على معلومات مهمة عن الإضافة،كإسم الإضافة،رقم الإصدار و وصف الإضافة و يمكن أن يحوي العديد من المعلومات الأخرى،يمكنك الإطلاع عليها من هنا. دعونا نأخذ http://goo.gl/ZlOd3F مثالا بسيط manifest.json :

```
{
    "name": "BrowserActionExtension",
    "version": "0.0.1",
    "manifest_version": 2,
    "browser_action": {
        "default_title": "That's the tool tip",
        "default_popup": "popup.html"
    }
}
```

سأشرح كل سطر على حدى:
name: و هو اسم الإضافة
version : إصدار الإضافة
manifest_version: هي إصدار "محرك" المتصفح،و ضعه دائما 2
browser_action:هي تعريف ببعض الخصائص
default_title: التسمية التي تظهر في المتصفح
default_popup: و تمثل ملف واجهة الإضافة

Background Files: و هي الملفات الأساسية و يمكن أن نعتبرها القلب النابض للإضافة و التي تحوي الأكواد البرمجية الأساسية لها،و تنقسم إلى قسمين:

persistent: و هي الأكواد الأساسية التي يتم تشغيلها بمجرد فتح المتصفح و تبقى في حالة العمل ما دام المتصفح شغالا

event:

و هي اكواد الأحداث و يتم تشغيلها عند حدث معين كأن يطلب المستخدم القيام بشيء معين من الإضافة.يتم تعريف كلا النوعين في ملف manifest.json: كالتالي

```
"background": {
    "scripts": ["background.js"],
    "persistent": false/true
}
```

Script :

Persistent تمثل مكان تواجد الملف الذي يحوي الأكواد أما فهي خيار منطقي إما مفعل أو غير مفعل،سنفصل في هته الجزئية في الدروس المقبلة بحول الله.

Content Script: و نعني بها في أي مكان سيتم تشغيل الإضافة،فهنالك إضافات لا تعمل إلا في موقع واحد مثل إضافة موقع ميجا الخاص بالرفع فهي لا تعمل إلا عندما تتصفح الموقع أو تقوم أنت بفتحها يدويا و هنالك إضافات تعمل مع جميع المواقع دون استثناء،يتم تعريف ذلك أيضا في ملف manifest.json: كالتالي

```
"content_scripts": [
    {
        "matches": ["http://*/*", "https://*/*"],
        "js": ["content.js"]
    }
]
```

خاصية Browser_action

،فقد ذكرت أنها تعريف لبعض الخصائص،نعم فهي تستخدم للعديد من الأمور منها تعيين أيقونة للإضافة،شاهد المثال التالي

```
"browser_action": {
   "default_icon": {
      "19": "icons/19x19.png",
      "38": "icons/38x38.png"
   },
   "default_title": "That's the tool tip",
   "default_popup": "popup.html"
}
```

بالطبع يتم تعريفها في ملف

manifest.json

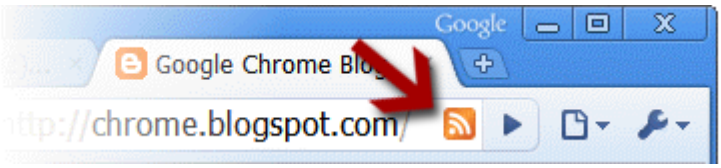
،أعتقد أن كل شيء واضح و لا يستدعي الشرح المستفيض له،الآن للنتقل لجزئية أخرى و هي التسمية التي

:فيتم إضافتها عن طريق السطر التالي في نفس الملف yeah تظهر مع أيقونة الإضافة و أقصد هنا الكتابة

```
chrome.browserAction.setBadgeText({text: "yeah"});
```

page_action هنالك خاصية أخرى يمكن إستخدامها في تطوير تطبيقات متصفح كروم و هي

كمثال عليها



و يتم تعريفها في نفس الملف السابق كالتالي:

```
"page_action": {
   "default_icon": {
      "19": "images/icon19.png",
      "38": "images/icon38.png"
   },
   "default_title": "Google Mail",
   "default_popup": "popup.html"
}
```

بصراحة هنالك العديد من الإضافات،و لا يمكنني بأي حال من الأحوال ذكرها كلها،و بالتالي إن كنت تريد معرفة المزيد من الخصائص فالأجدر بك زيارة موقع مطوري جوجل http://goo.gl/kA94MO من هنا.

الان سنتعرف على أشياء جديدة مهمة جدا تتعلق بهذا المتصفح

نكمل سويا ما بدأناه من مسيرة التعرف على بنية متصفح جوجل كروم،حتى نستطيع فيما بعد برمجة و تصميم إضافات خاصة بنا له...اليوم سنتعرف على أشياء جديدة مهمة جدا تتعلق بهذا المتصفح.

أدوات المطورين:Developer Tools

تمكنك هته الخاصية من إضافة ألسنة جديدة لتطبيقك و بالتالي يمكنك جعل الإضافة تعمل لأكثر من مرة،حتى تستطيع استخدام هته الخاصية عليك إنشاء صفحة ويب جديدة و بعدها تطلبها في ملف manifest.json كالآتي:

```
"devtools_page": "devtools.html"
```

في داخل الملف نقوم بتضمين ملف الأكواد البرمجية الخاصة بالجافاسكريبت كالتالي

كما تلاحظون علينا انشاء ملف جديد بإسم

devtools.js

يمكنكم تسميته كما تشاؤون و لكن عليكم تغيير الإسم عند القيام بتضمين الملف.في ملف الجافاسكريبت ضع الكود التالي

```
chrome.devtools.panels.create(
   "TheNameOfYourExtension",
   "img/icon16.png",
   "index.html",
   function() {

   }
);
```

أعتقد أن الكود بسيط و مفهوم و لا يحتاج لشرح دقيق،فلا شيء مبهم فيه...لننتقل إلى نقطة أخرى...:

ال-Omnibox:

متصفح جوجل كروم يتيح لنا إصدار أوامر كتابية له لتنفيذها،هذه الأوامر هي في الأصل كلمات مفتاحية تم وضعها من قبل مبرمجي الإضافات.

يتم تعريف الكلمات المفتاحية في ملف manifest.json و تتم العملية كالأتي:

```
"omnibox": { "keyword" : "yeah" }
```

تلاحظون بأنه تم تعريفomnibox على أنه مصفوفة(قاموس).يتم تعريف الكلمات المفتاحية في ملف الخلفية-Background تتذكرونه- كالآتي:

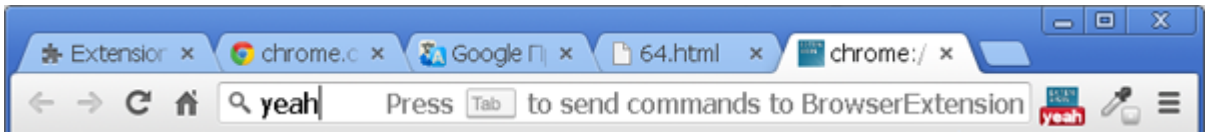
```
chrome.omnibox.onInputChanged.addListener(function(text, suggest) {
   suggest([
     {content: text + " one", description: "the first one"},
     {content: text + " number two", description: "the second entry"}
   ]);
});
```

```
chrome.omnibox.onInputEntered.addListener(function(text) {
   alert('You just typed "' + text + '"');
});
```

لو قمنا بكتابة

yeah : في المتصفح لتحصلنا على التالي

:



كيف حالكم مع متصفح جوجل كروم لحد الان ؟

بطبيعة الحال يوفر جوجل كروم العديد من الوظائف عن طريق الـ

API

Application Programming Interfaceلجملة اختصار فهي يعرفها لا لمن،

و هي تخص الوظائف التي يوفرها جوجل كروم أو،

. يستخدمها في عمله و يتيح للمطور استخدامها و الإستفادة منها في تطبيقاته

كما ذكرت قبل قليل،فإن كروم يوفر العديد من الوظائف و لا يمكن بأي حال من الأحوال تغطيتها جميعا و لكن عليك الإطلاع على التوثيق

.الرسمي لها حتى تستطيع فهم آلية العمل مع معرفة كيفية الإستخدام

يمكنك زيارة التوثيق الرسمي لها. من هنا http://goo.gl/SIlN3l

ننتقل إلى الجزء الثاني و هو الرسائل،تسمى رسائل و لكنها ليست كذلك.هي في الحقيقة أداة مراقبة لأداء الإضافة أو القيام بعمل ما استنادا
:لنتيجة معينة و هي نوعان

One-Time Request:

سأعطيكم مثالا بسيطا حتى تتضح الأمور،لو اردنا الإستفسار أو اخذ معلومات عن صفحة معينة،و نريد اخذ معلومات تخص

الـDOM :الخاص بالصفحة...سنقوم بكتابة الكود التالي

```
chrome.extension.onMessage.addListener(function(request, sender, sendResponse) {
   switch(request.type) {
      case "dom-loaded":
         alert(request.data.myProperty);
      break;
   }
   return true;
});
```

يتم كتابة هذا الكود في صفحة الـbackground

اعتقد أنكم تتذكرونها) و في ملف)

content

سنضع الكود التالي لاستقبال المعلومات

```
window.addEventListener("load", function() {
    chrome.extension.sendMessage({
        type: "dom-loaded",
        data: {
            myProperty: "value"
        }
    });
}, true);
```

Long-lived Connection:

يستخدم هذا النوع لإنشاء اتصال دائم،و ذلك عن طريق وضع الكود التالي في صفحة الـ content

```
var port = chrome.runtime.connect({name: "my-channel"});
port.postMessage({myProperty: "value"});
port.onMessage.addListener(function(msg) {
    // do some stuff here
});
```

و في صفحة الـbackground

نضع الكود التالي

```
chrome.runtime.onConnect.addListener(function(port) {
    if(port.name == "my-channel"){
        port.onMessage.addListener(function(msg) {
            // do some stuff here
        });
    }
});:
```

لم يتم شرحها بالتفصيل في الجانب التطبيقي لأنها أمور برمجية لن تستطيع فهمها إن لم تكن تمتلك خبرة بسيطة مع لغة الجافاسكريبت

www.intrprof.com