

MANUALE PER LO SVILUPPO DI UNA ESTENSIONE PER CHROME E FIREFOX

S3-Calendar 1.1

BUSCO PASQUALE
LUCENTE VINCENZO
TRENTADUE VITO

SOMMARIO

Sviluppo estensioni per Google Chrome	2
Struttura dell'estensione	2
Il manifest	4
Il file main.....	6
I file options	6
I file popup	12
Il file di background.....	13
Test dell'estensione	15
Installazione dell'estensione.....	17
Sviluppo estensioni per Mozilla Firefox.....	18
Installazione ADD-ON SDK	18
Struttura dell'estensione	20
Test dell'estensione	21
Installazione dell'estensione.....	24
S3-Calendar	25
Codice JavaScript di S3-Calendar	27
Commento del codice Javascript di S3-Calendar	31
Suggerimenti per gli sviluppatori.....	35

SVILUPPO ESTENSIONI PER GOOGLE CHROME

Google Chrome consente la creazione di estensioni che sono, in alcuni casi, applicazioni molto complesse. Per approcciare in modo semplice e veloce a questo mondo si può iniziare con il creare semplici estensioni che vadano a modificare il comportamento delle pagine Web, quelli che nel gergo delle applicazioni Chrome vengono detti Content Scripts. Realizzare tali Scripts è molto semplice, in quanto è possibile utilizzare il linguaggio Javascript e qualsiasi libreria ad esso associabile, come JQuery.

STRUTTURA DELL'ESTENSIONE

A differenza degli addon per Firefox, le estensioni per Chrome, presentano una struttura più semplice ed in generale più elastica. In queste estensioni infatti, il file principale, chiamato “manifest”, mappa tutte le risorse all'interno dell'estensione e lascia al progettista la libertà di organizzare i restanti file.

In generale un'estensione è composta dai seguenti file:

File	Descrizione
manifest.json	File fondamentale in cui si definiscono tutte le caratteristiche dell'estensione
main.js	Si possono avere uno o più file Javascript che definiscono la logica operativa dell'estensione
options.html	Pagina html con all'interno un form per permettere all'utente di personalizzare l'estensione tramite delle impostazioni
options.js	Direttamente collegata al file omonimo in html, permette di salvare, caricare e gestire le impostazioni dell'utente
popup.html	Nel caso di estensioni mostrate sulla toolbar del browser, questo file funge da interfaccia per le impostazioni del plug-in.
popup.js	Ha la stessa funzione del file “options.js”. I due file infatti possono coincidere.
immagine.png	L'estensione può contenere una o più immagini, che servano da icona, o da pulsanti, o per altri scopi.
immagine.png	L'estensione può contenere una o più immagini, che servano da icona, o da pulsanti, o per altri scopi.

background.js (html)	Questa pagina viene inizializzata all'avvio del browser e costituisce un'interfaccia di comunicazione tra le altre componenti dell'estensione.
jquery-2.1.4.min.js	Questo file è la libreria jQuery, che permetterà il recupero di informazioni da altre pagine web. Questo file necessita di essere caricato mediante il file manifest.json.

Di seguito saranno analizzati i singoli file componenti le estensioni, ed in particolare sarà mostrato il codice dei file scritti per il plug-in realizzato.

IL MANIFEST

Come già accennato, è nel manifest che si può segnalare al browser Chrome cosa farà l'applicazione, di quali file o librerie avrà bisogno per funzionare. Il manifest altro non è che un file in formato Json e le caratteristiche dell'applicazione vengono definite come coppie attributo-valore all'interno dell'oggetto.

JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati. Per le persone è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi. Si basa su un sottoinsieme del linguaggio di programmazione JavaScript, Standard ECMA-262 Terza Edizione - Dicembre 1999. JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.

Tornando al file "manifest", sarà mostrato di seguito un esempio esaustivo, e saranno illustrate tutte le caratteristiche dello stesso. Prima però, è doveroso fare una precisazione. Tutto il contenuto dell'estensione deve essere collocato in una directory che al suo interno potrà avere un numero indefinito di sub-directory, ma è importante che il manifest, sia nella root directory e mappi i percorsi di tutte le risorse necessarie.

```
1. {
2.   "name": "Nome estensione",
3.   "version": "Versione dell'estensione",
4.   "manifest_version": 2,
5.   "description": "Descrizione dell'estensione",
6.   "options_page": "options.html",
7.   "content_security_policy": "script-src 'self' https://example.com; object-src 'self'",
8.   "icons": { "256": "images/immagine.png" },
9.   "permissions": [ "tabs", "storage", "resource", "background", "notifications" ],
10.  "background": { "scripts": ["background.js"] "persistent": true},
11.  "web_accessible_resources": [{"images/icon.png", "images/immagine.png"}],
12.  "permissions": ["https://*.example.com/*"],
13.  "browser_action": {"default_icon":
14.    {
15.      "19": "images/icon19.png",
16.      "38": "images/icon38.png"
17.    },
18.    "default_popup": "popup.html"
19.  },
20.  "content_scripts": [ { "matches": ["https://*.example.com /*"], "js": [ "main.js" ]
    , "css": [ "stili.css" ] } ]
20. }
```

Si illustra di seguito il significato di ogni attributo del file (l'asterisco indica l'obbligatorietà dello stesso):

- **name***: una stringa che indica il nome scelto per l'estensione;
- **version***: una stringa che indica il numero di versione dell'estensione;
- **manifest_version***: questo numero indica la versione del file manifest, che si intende utilizzare. Attualmente lo standard di Chrome indica, l'utilizzo della versione "2". La differenza tra le diverse versioni, risiede essenzialmente nelle policy di sicurezza e nelle risorse ed API, messe a disposizione dal browser;
- **description***: descrizione testuale delle caratteristiche dell'estensione;
- **option_page**: indica quale pagina HTML implementa il form per il settaggio dell'estensione;
- **content_security_policy**: Google Chrome implementa delle policy di sicurezza per le estensioni. Utilizzando un manifest di versione 2, implicitamente si applicano all'estensione dei divieti sull'esecuzione di scripts inline all'interno dei file che compongono l'estensione stessa. Ad esempio se nel file options.html si provasse ad inserire uno script racchiuso nel tag `<script></script>`, Chrome solleverà un errore all'atto dell'installazione del plug-in. Si può ovviare a tale problematica, inserendo il link ad uno script esterno (`<script type="Javascript" src="codice.js"></script>`). Se tale script, dovesse trovarsi al di fuori del pacchetto dell'estensione, bisogna specificare nel "content_security_policy", l'URL dello script ed i permessi associati. Maggiori dettagli sui permessi sono disponibili nella documentazione dell'estensioni fornita da Google;
- **icons**: associa un'immagine identificativa, all'estensione. In particolare bisogna specificare, la dimensione dell'immagine ("256") ed il percorso della stessa all'interno del package;
- **permission***: fornisce diversi tipi di permessi all'estensione. Nell'esempio sono stati specificati i seguenti:
 - **tabs**: permette all'estensione di aprire e chiudere schede sul browser
 - **storage**: permette di memorizzare file, o dati all'interno delle pagine HTML. Tale permesso è utile ad esempio, quando si vuole utilizzare la direttiva HTML5 "localStorage". Questa istruzione, permette ad una pagina HTML di memorizzare dei dati in maniera persistente. Il browser associa tali dati all'estensione ed ad ogni riavvio dello stesso, essi saranno sempre disponibili. I dati vengono persi se l'estensione viene rimossa;
 - **resource**: possibilità di accesso a risorse esterne;
 - **background**: consente l'esecuzione dello script di background di cui si parlerà in questo documento;
 - **notifications**: permette all'estensione di fornire notifiche all'utente;
 - **http://*.example.com/***: indica a quale dominio sono associati i permessi specificati;

- **background:** indica la presenza di una pagina di background sempre attiva all'interno del browser. Il valore di questo attributo è un file con estensione html o js. La coppia attributo – valore, "persistent: true", indica appunto il carattere persistente della pagina di background;
- **web_accessible_resources:** specifica quali risorse all'interno del package, possono essere referenziate da pagine web. Ad esempio se si ha la necessità di modificare il DOM di una pagina web, inserendo un'immagine presente all'interno dell'estensione, bisogna specificare che tale immagine è accessibile dall'esterno;
- **browser_action:** i valori di questo attributo, specificano una serie di direttive per il browser. Come nell'esempio, è possibile specificare se visualizzare un'icona per l'estensione sulla toolbar ed è possibile anche associarvi un menù di tipo popup;
- **content_scripts:** questo è forse l'attributo più importante. Al suo interno vanno elencati, tutti i file Javascript che saranno eseguiti dall'estensione e ad ognuno è possibile associare un array di "matches", ossia uno o più domini, sulle cui pagine il codice Javascript, verrà iniettato. In questo attributo inoltre, è possibile specificare, eventuali fogli di stile CSS necessari all'estensione.


IL FILE MAIN

Non vi sono direttive sul nome di questo file, si può quindi associarvi qualsiasi nome seguito dall'estensione js. Qualsiasi nome scelto dovrà però, essere specificato nel file manifest. Ogni estensione per Chrome, può essere composta da più file js che ne realizzano la logica operativa. Almeno un file di questo tipo deve essere sempre presente, e per ogni file js che agirà su una pagina di un dominio, bisogna specificare nel manifest, il percorso di tale file ed il dominio sul quale andrà in esecuzione. Come già detto ciò viene realizzato con gli attributi "content_scripts" e "matches". In particolare il codice di ogni file specificato in questo modo, verrà iniettato nelle pagine web specificate (tramite URL) all'interno dell'array di "matches", mentre nell'array "js" nel nostro caso saranno presenti il file "main.js" e "jquery-2.1.4.min.js".

I FILE OPTIONS

I file options sono solitamente due, uno in formato HTML e l'altro Javascript. Questi file vengono utilizzati, in caso di estensioni che consentono il settaggio di alcuni parametri o semplicemente la personalizzazione grafica dell'estensione stessa. Il file in formato HTML mostra un'interfaccia grafica con cui permettere all'utente di impostare le proprie preferenze, mentre il file Javascript implementa i metodi per il salvataggio ed il caricamento delle stesse. In quest'ultimo file ovviamente, possono risiedere funzioni di diversa natura. I due file vengono separati per via delle Content Security Policy di Chrome, che non consentono l'esecuzione di script inline.

Di seguito viene mostrato uno screenshot della pagina di impostazioni realizzata per l'estensione S3-Calendar, seguita dai codici HTML e Javascript.

Google calendar

My Calendar: Save

1. Open your [Google Calendar](#) page
2. Click on "my calendar" on the left
3. Click on the arrow, a popup menu will appear
4. Click on "calendar settings"
5. Click on your favourite calendar
6. Copy from "calendar address" row the "calendar ID"
7. ID appears like this:
abcde789caezt989@group.calendar.google.com

PV Solution © 2014

Il codice HTML, ed il CSS associato, vengono riportati di seguito.

```
1. <html>
2. <head>
3. <style>
4.         #content {
5.             background-color: white;
6.             border: 4px solid #B5C7DE;
7.             border-radius: 12px;
8.             margin: 40px auto 20px;
9.             padding: 8px;
10.            width: 600px;
11.            font-family: Verdana;
12.        }
13.        .option_row {
14.            clear: left;
15.            padding: 2.5em 1em 0;
16.            text-align:center
17.        }
18.
```

```
19. #status {
20.     background-color: rgb(255, 241, 168);
21.     display: none;
22.     margin-left: 3px;
23.     padding: 1px 2px;
24.     text-align: center;
25.     font-size: 15px;
26.     color: #000;
27. }
28. #multiCalendarText {
29.     font-size: 15px;
30.     color: #000;
31. }
32. body {
33.     background-color: "#ebeff9";
34. }
35. #logo {
36.     font-size: 20px;
37.     text-align: center;
38. }
39. #extensionName {
40.     color: #444;
41. }
42. .save {
43.     background:-webkit-gradient( linear, left top, left bottom, color-stop(0.05,
44. #79bbff), color-stop(1, #378de5) );
45.     background:-moz-linear-gradient( center top, #79bbff 5%, #378de5 100% );
46.     filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#79bbff',
47. endColorstr='#378de5');
48.     background-color:#79bbff;
49.     border-radius:0px;
50.     text-indent:0;
51.     border:1px solid #84bbf3;
52.     display:inline-block;
53.     color:#ffffff;
54.     font-family:Verdana;
55.     font-size:15px;
56.     font-weight:bold;
57.     font-style:italic;
58.     height:35px;
59.     line-height:15px;
60.     width:80px;
61.     text-decoration:none;
62.     text-align:center;
63.     text-shadow:1px 1px 0px #528ecc;
64. }
65. .save:hover {
66.     background:-webkit-gradient( linear, left top, left bottom, color-stop(0.05,
67. #378de5), color-stop(1, #79bbff) );
68.     background:-moz-linear-gradient( center top, #378de5 5%, #79bbff 100% );
69.     filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#378de5',
70. endColorstr='#79bbff');
71.     background-color:#378de5;
72. }.
73. save:active {
74.     position:relative;
75.     top:1px;
76. }
77. input {
```

```
130.         <tr>
131.             <td colspan='3' style="color: #b5c7de; text-align: right; font-size:
132.                 11px;" >PV Solution &copy; 2014
133.             </td>
134.         </tr>
135.     </tbody>
136. </table>
137. </div>
138. </div>
139. </body>
140. <script type="text/javascript" src="options.js"></script>
141. </html>
```

Il codice Javascript, che si occupa del salvataggio del calendario preferito e del suo caricamento all'avvio del browser, viene riportato di seguito:

```
1. function save_options() {
2.     var select = document.getElementById('calendar');
3.     var calendar = select.value;
4.     localStorage["calendar_name"]=calendar;
5.     var status = document.getElementById("feedback");
6.     status.innerHTML = "Options Saved!!!";
7.     var bkg = chrome.extension.getBackgroundPage();
8.     bkg.settings.foo = calendar;
9. }
10. function restore_options() {
11.     var favorite = localStorage["calendar_name"];
12.     var input = document.getElementById("calendar");
13.     input.value = favorite;
14.     bkg.settings.foo = favorite;
15. }
16. document.addEventListener('DOMContentLoaded', restore_options);
17. document.querySelector('#save').addEventListener('click', save_options);
```

Questo script viene invocato dalla pagina HTML, ed è interessante notare come si rispettino le policy di sicurezza imposte da Chrome. Sul tasto “Save”, realizzato in HTML, non vi è alcun evento di tipo “onclick”, ma nel file JS vi sono due event listner che catturano il click e provvedono a modificare il DOM della pagina HTML. In questo modo non vi è codice inline nelle file di markup, ma si ottiene lo stesso comportamento desiderato. Nel file JS invece è interessante notare come vengono memorizzate le preferenze dell’utente, tramite la direttiva HTML5 localStorage, a cui si è già fatto riferimento in questo documento. Inoltre, quello che viene memorizzato, viene anche passato alla pagina di background che resta sempre in ascolto. In questo modo è possibile comunicare, tramite dei messaggi, con i content script iniettati nelle pagine web.

I FILE POPUP

I file popup.html e popup.js hanno funzioni simili ai file “options”. In particolare tramite questi file è possibile visualizzare un menu a cascata che si apre al click sull'icona dell'estensione, visualizzata sulla toolbar del browser. Ovviamente la scelta di implementare tali file è facoltativa. Nel caso dell'estensione “S3 – Calendar”, non è stato implementato alcun popup, in quanto l'icona del plug-in visualizzata sulla toolbar risulta invasiva e soprattutto superflua visti gli obiettivi dell'estensione realizzata.



La figura mostra un esempio di finestra di popup realizzato utilizzando semplice codice HTML e CSS. Inoltre è possibile associare a tale finestra delle azioni tramite uno script. I file HTML e Javascript saranno collegati come mostrato per i file options, ovvero rispettando le Content Security Policy di Google Chrome.

IL FILE DI BACKGROUND

Questo file è realizzabile sia in formato HTML che in formato Javascript, tuttavia solitamente il file in formato HTML contiene al suo interno codice Javascript e un HTML molto povero, pertanto si preferisce creare file direttamente in formato JS. La pagina di background ha la particolarità di essere eseguita all'avvio del browser e restare sempre attiva per tutta la sessione di lavoro, è possibile pertanto paragonarla ad un demone. Sebbene tale file, possa essere utilizzato per i più svariati scopi, esso viene spesso utilizzato per fornire un meccanismo di comunicazione tra il codice iniettato dei file "content_script", o per far comunicare questi ultimi con i file delle opzioni. Nel caso dell'estensione "S3-Calendar", il file di background svolge appunto, quest'ultimo compito. Di seguito la sua realizzazione.

```
1. settings = {
2.   get foo() {
3.       return localStorage['foo'];
4.   },
5.   set foo(val) {
6.       localStorage['foo'] = val;
7.   }
8. };
9.
10. chrome.runtime.onMessage.addListener(
11.   function(request, sender, sendResponse) {
12.       console.log(sender.tab ? "from a content script:" +
13.         sender.tab.url : "from the extension");
14.       if (request.greeting == "calendar")
15.         sendResponse({farewell: settings.foo});
16.   }
17. );
```

La variabile "setting" viene inizializzata dalla pagina delle impostazioni che invia una richiesta al browser per ottenere uno stream di comunicazione con la pagina di background. Questo viene realizzato come segue:

```
1. var bkg = chrome.extension.getBackgroundPage();
2. bkg.settings.foo = calendar;
```

Con l'istruzione `"chrome.extension.getBackgroundPage();"` apre lo stream di comunicazione, mentre con l'istruzione `"bkg.setting.foo = calendar"`, è possibile assegnare il valore della variabile `"calendar"` alla variabile `"setting"` della pagina di background.

Il blocco di codice successivo invece, imposta un listener sull'invio di messaggi. In particolare se un qualsiasi codice Javascript, appartenente all'estensione, invia un messaggio di richiesta, la pagina di background lo intercetta, individua il tipo di richiesta (`"if (request.greeting == 'calendar')"`) e invia la risposta desiderata (`"sendResponse({farewell: settings.foo});"`).

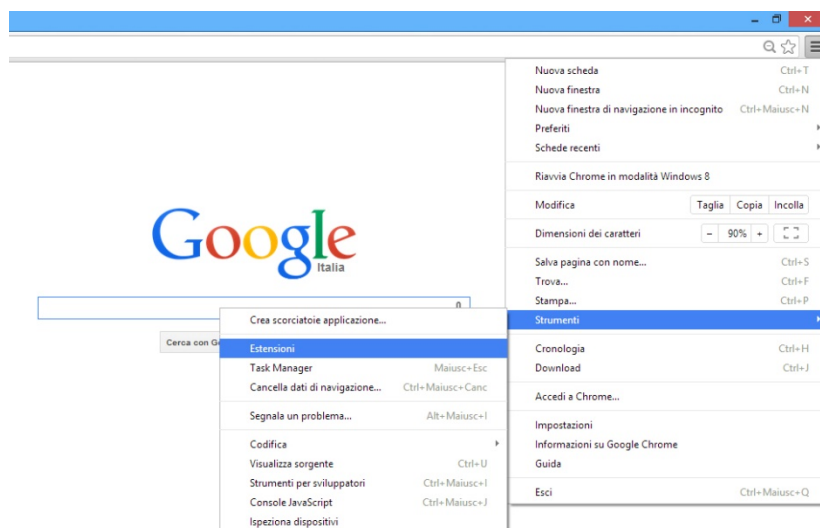
Nello script da iniettare nelle pagine web, ci sarà quindi una funzione che invia il messaggio di richiesta che a sua volta, come già detto, sarà preso in carico dalla background page.

```
1. chrome.runtime.sendMessage({greeting: "calendar"}, function(response) {
2.     calendar_name = response.farewell;
3. });
```

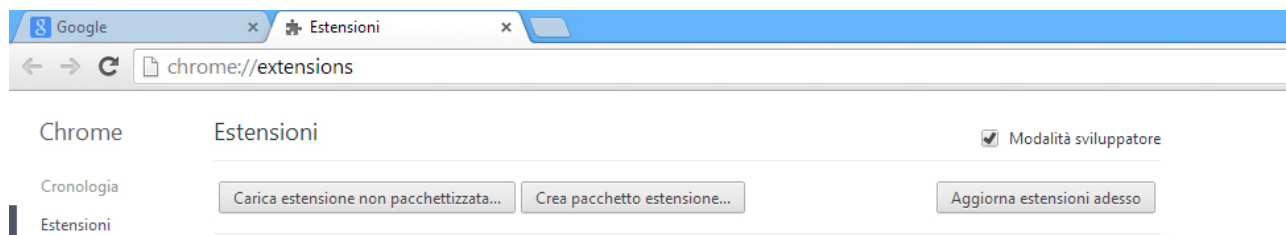
Come si può notare la funzione invia una richiesta di tipo `"calendar"`, e la pagina di background risponde in base a quanto richiesto. Un aspetto importante dello scambio di messaggi tra i codici dell'estensione, sta nel fatto che questo avviene in modo asincrono. Mentre la pagina di background è sempre in ascolto, la funzione che invia il messaggio di richiesta, viene istanziata ed attivata solo al momento dell'iniezione del codice Javascript all'interno della pagina web. Questo aspetto è da tenere in considerazione quando si invia un messaggio di richiesta. Nel caso dell'estensione `"S3-Calendar"` ad esempio, in un primo momento si inviava la richiesta del valore della variabile `"calendar"`, impostata dall'utente, nello stesso momento in cui la funzione, che la utilizza (`"addToGoogleCalendar"`), veniva invocata. Il risultato era che la variabile `"calendar"` risultava sempre vuota, in quanto il codice veniva iniettato prima di essere invocato. Il problema è stato risolto, istanziando una variabile globale all'interno del metodo `"main"` che è il primo ad essere iniettato ed invocato.

TEST DELL'ESTENSIONE

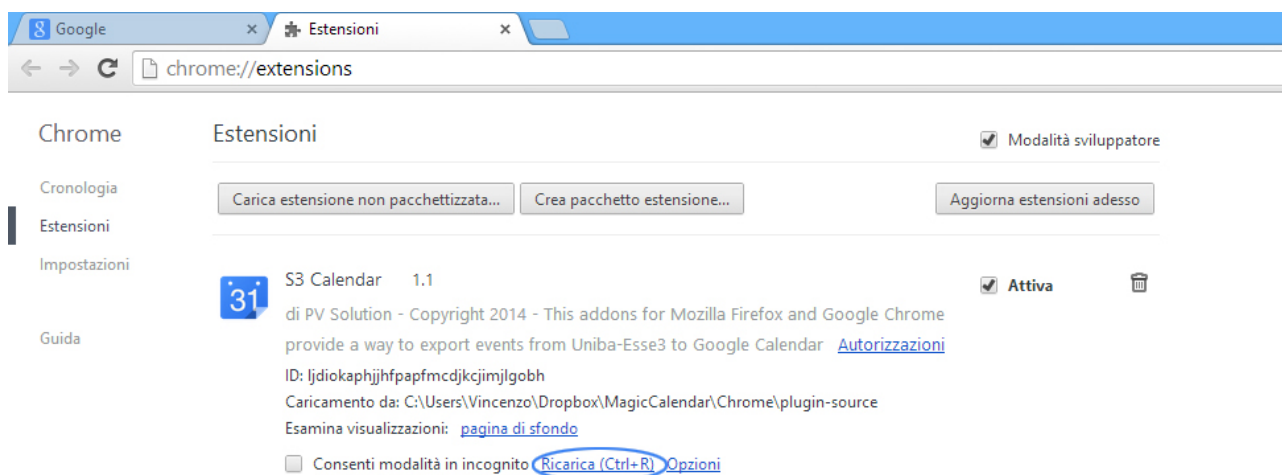
Google Chrome offre un modo molto semplice di testare le proprie estensioni. Come detto in precedenza è sufficiente inserire tutti i file, di cui sopra, all'interno di una stessa cartella radice. Fatto questo bisogna aprire il browser e dall'icona nell'angolo a destra sulla toolbar, selezionare strumenti, poi estensioni.



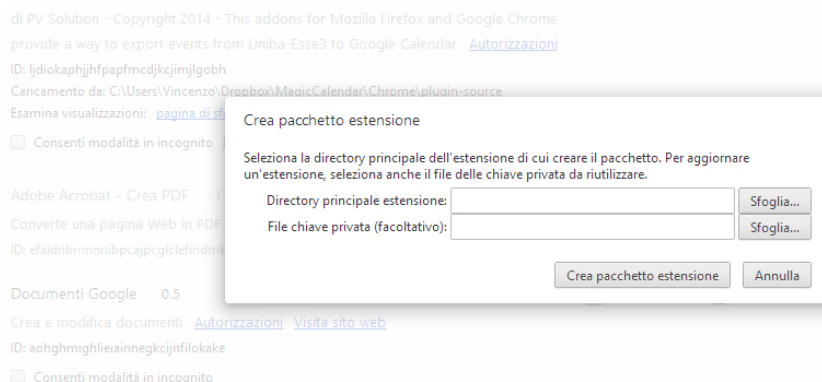
Nella finestra delle estensioni, spuntare l'opzione "Modalità sviluppatore" ed apparirà un tasto con la scritta "Carica estensione non pacchettizzata". Selezionare la directory dell'estensione e se tutto è stato scritto in maniera corretta, sarà possibile testarla sul browser.



Quando si apporteranno delle modifiche al plug-in, non ci sarà bisogno di rifare questa operazione, ma sempre nella pagina delle estensioni, basterà cercare la propria e cliccare su "Ricarica".

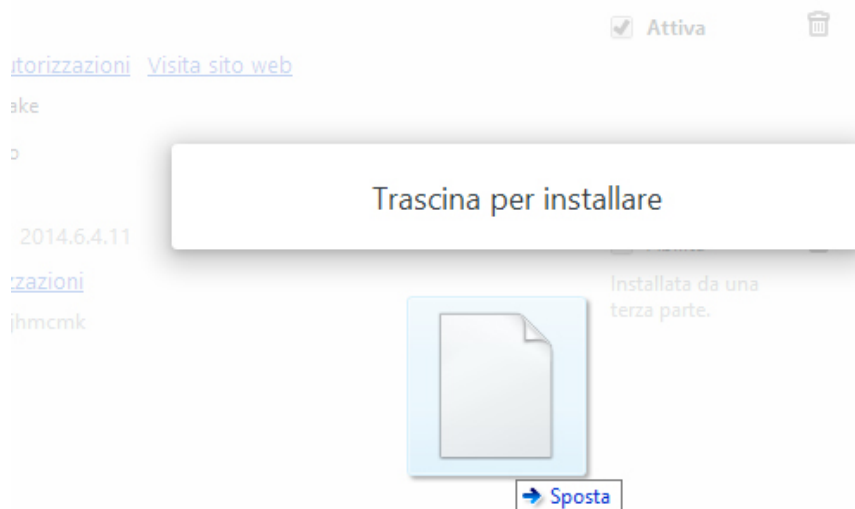


Infine quando l'estensione è ultimata, ancora una volta dal pannello delle estensioni, cliccare su “Crea pacchetto estensione”, selezionare la directory corrispondente e confermare. Saranno creati, nello stesso percorso della directory selezionata, un file con estensione “crx” ed uno con estensione “pem”. Il primo è il pacchetto installabile, contenente l'estensione, il secondo è un certificato associato all'estensione.

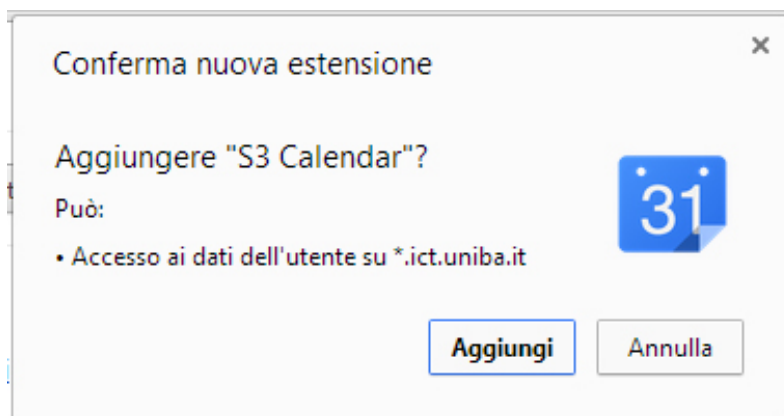


INSTALLAZIONE DELL'ESTENSIONE

Nella toolbar di Chrome, cliccare sull'icona nell'angolo a destra. Dal menu in popup, selezionare strumenti e poi estensioni quindi, trascinare il file con estensione "crx" sulla finestra che si aprirà.



Cliccare infine su "Aggiungi" nel popup che apparirà dopo aver trascinato il file "crx" sul pannello delle estensioni.



SVILUPPO ESTENSIONI PER MOZILLA FIREFOX

Prima dell'avvento dei browser basati su Gecko 2.0 lo sviluppo di estensioni per Firefox poteva avvenire solo in due modi diversi:

- Attraverso l'utilizzo di XUL, un overlays specifico per interfacce
- Oppure utilizzando API e moduli Javascript per interagire con le applicazioni e pagine web

Ora invece altre due tecniche alternative sono disponibili per lo sviluppatore di estensioni, ovvero:

- L'utilizzo dell'Add-on SDK
- Oppure le estensioni "restartless"

Entrambe queste nuove tecniche non utilizzano più overlays XUL, ormai considerato obsoleto, ma utilizzano un più efficiente e sicuro framework sviluppato da Mozilla.

INSTALLAZIONE ADD-ON SDK

Per sviluppare in ambiente Windows un'estensione Firefox basata su Add-on SDK è necessario soddisfare tre prerequisiti software:

- Installare Python 2.6 (attenzione la v2.7 o superiori, non sono compatibili) nella cartella di installazione predefinita (C:\Python26)
- Installare il browser Firefox
- Scaricare ed estrarre Add-on SDK sul proprio pc (C:\addon-sdk)

Per procedere alla creazione di un'estensione per Firefox è necessario aprire la riga di comando digitando

```
1. cmd
```

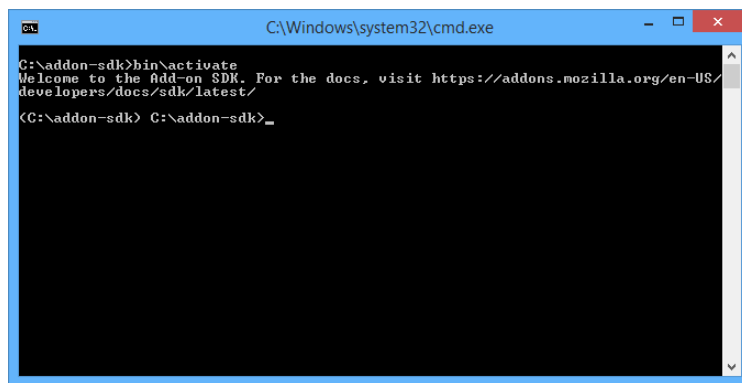
Successivamente spostarsi nel percorso dove è stato estratto precedentemente l'SDK, nel nostro caso "C:\addon-sdk", utilizzando il comando

```
1. cd addon-sdk
```

Infine per avviare l'SDK digitare questo comando

```
1. bin\activate
```

Se tutto è eseguito correttamente si avrà di fronte ad una schermata simile a quella riportata nella figura seguente. Da questo momento in poi tutti i comandi digitati verranno interpretati dall'SDK e successivamente eseguiti.



Uno dei vantaggi dell'utilizzo dell'SDK è la possibilità di automatizzare una serie di operazioni che altrimenti risulterebbero complesse e spesso noiose. Infatti per realizzare un'estensione che venga riconosciuta dal browser è necessario creare correttamente una precisa struttura di cartelle e file. Digitato i seguenti comandi è possibile effettuare velocemente tutte queste operazioni.

```
1. mkdir my-addon
2. cd my-addon
3. cfx init
```

Il primo comando è un comando tipico del DOS e permette di creare una nuova cartella nella directory di lavoro corrente. A seguire tramite il comando CD (current directory) ci si sposta nella cartella appena creata e l'ultimo comando ci permette di generare una nuova estensione. È importante sottolineare che nel caso di una estensione già esistente è sufficiente digitare solo il secondo comando.

Se tutto è andato a buon fine si avrà di fronte ad una schermata molto simile a quella della figura seguente dove vengono descritte le operazioni effettuate dall'SDK.

```
C:\Windows\system32\cmd.exe

C:\addon-sdk>bin\activate
Welcome to the Add-on SDK. For the docs, visit https://addons.mozilla.org/en-US/
developers/docs/sdk/latest/

(C:\addon-sdk) C:\addon-sdk>mkdir my-addon

(C:\addon-sdk) C:\addon-sdk>cd my-addon

(C:\addon-sdk) C:\addon-sdk\my-addon>cfx init
* lib directory created
* data directory created
* test directory created
* doc directory created
* README.md written
* generated jid automatically: jid1-hlk8bypJyBSe4g
* package.json written
* test/test-main.js written
* lib/main.js written
* doc/main.md written

Your sample add-on is now ready.
Do "cfx test" to test it and "cfx run" to try it. Have fun!

(C:\addon-sdk) C:\addon-sdk\my-addon>
```

STRUTTURA DELL'ESTENSIONE

Durante il processo precedente sono stati generati diverse cartelle contenenti alcuni file. Andiamo ora a vedere nel dettaglio a cosa servono e come vanno modificati per poter personalizzare l'estensione appena creata.

- **Directory: data**

In questa directory vanno posizionati tutti i file esterni che verranno inclusi nell'estensione. Per esempio è consigliabile inserire in questa cartella tutti gli script in formato JS, le immagini utilizzate e tutti quegli altri file che dovranno essere richiamati durante l'esecuzione. Nel nostro caso sono presenti i file "calendar.js" e "jquery-2.1.4.min.js".

- **Directory: doc**

Inserire in questa posizione tutta la documentazione relativa ai file creati. I file dovranno avere lo stesso nome del file che si vuole descrivere ma dovrà essere utilizzata l'estensione MD. Ad esempio per allegare la documentazione del file main.js posizionare in questa cartella un file chiamato main.md

- **Directory: lib**

È la directory più importante in quando i file principali vengono collocati qui. Di solito oltre al file principale main.js vengono inserite delle librerie esterne necessarie al funzionamento di alcuni plugin.

- **File: main.js**

All'avvio del browser vengono caricate tutte le estensioni che sono state installate e se non diversamente specificato questo sarà il primo file che viene eseguito. Qui si trovano le direttive principali, le require dei vari componenti dell'sdk e la definizione della variabili con scope globale. In questo file vanno specificati i file che devono essere utilizzati: `contentScriptFile: [data.url("jquery-2.1.4.min.js"), data.url("calendar.js")]`.

- **File: package.json**

Questo file si occupa di definire tutte le proprietà e le caratteristiche dell'estensione. È quindi paragonabile al file manifest di Chrome. Tra i tanti attributi che è possibile definire appartenenti al formato standard json troviamo:

- name -> nome interno dell'estensione
- title -> titolo dell'estensione visualizzato in Firefox
- id -> id univoco dell'estensione
- description -> descrizione formale dell'estensione da visualizzare nel browser
- author -> autore dell'estensione
- license -> tipo di licenza sotto quale è stata rilasciata l'applicazione
- version -> versione dell'estensione
- icon -> percorso relativo dell'icona che verrà utilizzata nell'estensione.

Inoltre in questo file è possibile specificare tutta una serie di campi che permettono la creazione delle preferenze utente.

- **File: README.md**

Per finire in questo file come per quelli contenuti nella cartella doc è possibile specificare in maniera testuale una serie di informazioni e disclaimer relativi all'estensione.

TEST DELL'ESTENSIONE

Una volta personalizzata l'estensione con il proprio codice è possibile compilarla e mandarla in esecuzione tramite un unico semplice comando

```
1. cfx run
```

Il browser si occuperà ogni volta che il comando viene eseguito di creare un nuovo profilo utente in modo da non interagire con sessioni già aperte, testare e infine pubblicare l'estensione. In questo modo le operazioni di debug sono molto facilitate in quanto sarà possibile leggere direttamente nella console di Windows tutti gli eventuali errori e warning presenti.

Infine quando si è sicuri che l'estensione creata sia funzionante e senza errori è possibile esportarla nel formato installabile XPI utilizzando il comando

```
1. cfx xpi
```

A seguire il codice commentato del file **main.js** dell'estensione S3-Calendar

```
1. var self = require("sdk/self");
2. var data = require("sdk/self").data;
3. var pageMod = require("sdk/page-mod");
4.
5. pageMod.PageMod({
6.   include: "*.ict.uniba.it",
7.   contentScriptFile: [data.url("calendar.js"), data.url("jquery-2.1.4.min.js")],
8.   contentScriptOptions: {
9.     img: data.url("calendar.png"),
10.    calendar: pref.prefs['calendarName']
11.  }
12. });
```

Le righe dalla 1 alla 3 definiscono delle variabili che fanno riferimento a tre moduli distinti dell'SDK. In particolare al rigo 3 la variabile `pageMod` di occupa di richiamare il componente `page-mod` indispensabile per poter modificare una pagina web aperta in Firefox.

Dalla riga 5 alla 12 vengono definite le caratteristiche della variabile `pageMod`.

- Al rigo 6 troviamo il dominio sul quale la variabile va ad agire. Infatti specificando “*.ict.uniba.it” indichiamo che lo script dovrà essere eseguito su tutte quelle pagine che nell’URL contengono la stringa indicata.
- Al rigo 7 tramite l’istruzione `contentScriptFile` definiamo i file che dovranno essere iniettati nella pagina sopra definita.
- Al rigo 8 `contentScriptOptions` permette di passare alcuni parametri allo script. Questa operazione si rende necessaria in quanto Firefox applica un isolamento tra i vari livelli del browser e non permette al codice in esecuzione in una pagina di comunicare direttamente con il codice dell’estensione.
 - Al rigo 9 viene definita la variabile `img` e gli viene assegnato il percorso interno dell’icona del calendario che viene visualizzata nella pagina web
 - Al rigo 10 si passa allo script il nome del calendario predefinito da utilizzare che l’utente ha definito nelle opzioni dell’estensione.

Di seguito invece il codice commentato del file **package.json** dell'estensione S3-Calendar

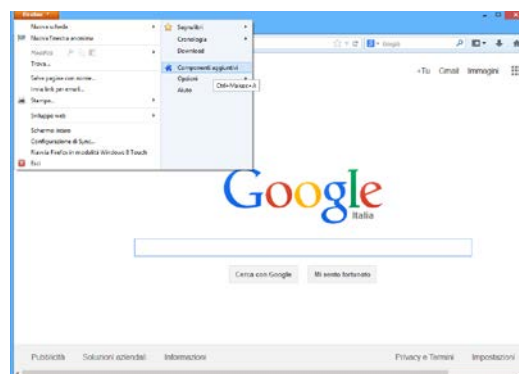
```
1. {
2.   "name": "s3calendar",
3.   "title": "S3-Calendar",
4.   "id": "jid1-ovVO3xmOZD0hHw",
5.   "description": "This is addons for Mozilla Firefox",
6.   "author": "PV Solution - Copyright 2014",
7.   "license": "MPL 2.0",
8.   "version": "1.1",
9.   "icon": "icon.png",
10.
11.   "preferences": [{
12.     "name": "calendarName",
13.     "title": "Goolge Calendar ID",
14.     "description": "TIPS",
15.     "type": "string",
16.     "value": ""
17.   }]
18. }
```

Dalla riga 2 alla riga 9 vengono specificate tutte le proprietà già viste precedentemente per il file package.json, invece dalla riga 11 alla 17 vengono specificati i parametri necessari alla creazione delle preferenze utente, in particolare:

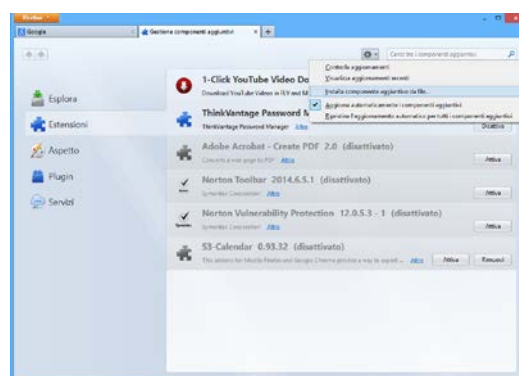
- name -> nome interno della preferenza
- title -> titolo visualizzato in Firefox
- description -> descrizione della preferenza per l'utente
- type -> tipo della preferenza ad esempio string, bool, integer
- value -> valore predefinito della preferenza

INSTALLAZIONE DELL'ESTENSIONE

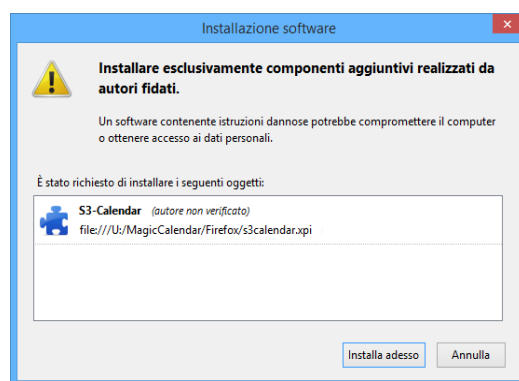
1. Aprire il browser Mozilla Firefox
2. Cliccare in alto sul tasto Firefox
3. Cliccare su Componenti Aggiuntivi



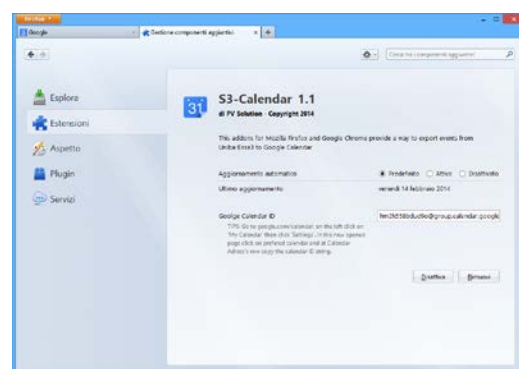
4. Cliccare sulla destra sul tastino a forma di ingranaggio
5. Cliccare su Installa componente aggiuntivo da file
6. Selezione il file dell'estensione in formato XPI dal proprio computer



7. Firefox visualizzare una schermata di riepilogo dell'estensione che si vuole installare, cliccare quindi su Installa adesso



8. Una volta installata l'estensione è possibile personalizzarla cliccando sull'apposito pulsante opzioni



S3-CALENDAR

L'estensione "S3-Calendar" ha l'obiettivo di fornire un'interfaccia tra la piattaforma "Esse3" dell'Università degli Studi di Bari Aldo Moro, e l'applicazione Google Calendar. In particolare l'estensione carpisce tutti i dati relativi ad un appello d'esame e crea un evento all'interno di Google Calendar alla data specificata, inserendo una serie di dettagli relativi all'esame. Questo è utile per tener traccia degli impegni universitari sia dei docenti che degli studenti, ma soprattutto è vantaggioso quando altre applicazioni come il calendario di "Android" o di "Outlook" vengono sincronizzate con il calendario di Google. In questo modo si può avere la propria agenda virtuale disponibile su qualsiasi dispositivo (pc, smartphone, tablet, ecc.) sempre ed ovunque.

Il codice sorgente del progetto è disponibile su GitHub al seguente indirizzo:


<https://github.com/s3calendar/s3calendar>

L'estensione lavora sul DOM di "Esse3" e visualizza accanto ad ogni appello un'icona a forma di calendario.

PROGETTAZIONE E PRODUZIONI DI CONTENUTI DIGITALI - [061670] - TERZO APPELLO 2013-2014								
Numero Iscrizione: 4 su 7								
Tipo Prova: orale								
Giorno	Ora	Edificio	Aula	Riservato per	Docenti Cognome Nome	Cancella	Stampa	Calendario
11/02/2014	10:00	Dipartimento di Informatica (Campus)		Nessun partizionamento	GENTILE ENRICHETTA			

SISTEMI PER LA COLLABORAZIONE IN RETE - [008051] - appello febbraio 2014								
Numero Iscrizione: 4 su 6								
Giorno	Ora	Edificio	Aula	Riservato per	Docenti Cognome Nome	Cancella	Stampa	Calendario
17/02/2014	17:00	c/o Studio del Docente		Nessun partizionamento	LANUBILE FILIPPO CALEFATO FABIO			

Cliccando su questa icona si aprirà un pagina di Google Calendar contenente tutte le informazioni sul appello. Basterà cliccare su “Salva” per averla nel proprio calendario. Inoltre per gli utenti che hanno più calendari Google, è stata prevista la possibilità di specificare quello preferito, nelle impostazioni del plug-in. In questo modo l’evento sarà sempre creato nel calendario specificato.



☐ Tutto il giorno
 ☐ Ripeti...

Dove

mappa

Videochiamata

Calendario

Creato da

Descrizione

Colore evento

☒
☐
☐
☐
☐
☐
☐
☐
☐
☐

Promemoria

Nessun promemoria impostato

Imposta il mio stato su

☐ Disponibile
 ☒ Occupato

Aggiungi invitati

Gli invitati possono

☐ modificare l'evento
 ☒ invitare altri
 ☒ vedere l'elenco invitati

CODICE JAVASCRIPT DI S3-CALENDAR

In questa sezione viene mostrato il codice Javascript dell'estensione S3-Calendar. L'implementazione per Chrome e Firefox è molto simile con alcune piccole differenze che sono state evidenziate nel testo. Infatti il **rosso** è impiegato per il codice Javascript utilizzato solo nella versione Chrome, in **verde** il codice esclusivo per Firefox.

```
1. var studentPage =
   "https://www.studenti.ict.uniba.it/esse3/auth/studente/Appelli/BachecaPrenotazion
   i";
2. var docentPage =
   "https://www.studenti.ict.uniba.it/esse3/auth/docente/CalendarioEsami/ElencoAppel
   liCalEsa";
3. var calendar_name="";
4.
5. doAction = function(i, r) {
6.     addToGoogleCalendar(getInfo(i, r));
7. };
8.
9. createInput = function(i, r) {
10.     var imageUrl = self.options.img;
11.     var imageUrl = chrome.extension.getURL("/icon.png");
12.
13.     var button = "<img id='googleCalendar'+i+' ' name='imageField' alt='Aggiungi a
   Calendar' title='Aggiungi a Calendar' " + "style='cursor: pointer;' src='" +
   imageUrl + "'>";
14.     document.getElementById('calendar'+i).innerHTML += button;
15.     document.getElementById('googleCalendar'+i).addEventListener("mouseup",
   function(event) {
16.         doAction(i, r);
17.     }, false);
18.
19. };
20.
21. main = function() {
22.
23.     chrome.runtime.sendMessage({greeting: "calendar"}, function(response) {
24.         calendar_name = response.farewell;
25.     }
26.
27.     if (document.URL.indexOf(studentPage) != -1) {
28.         var table_class_s = document.querySelectorAll('table.detail_table');
29.         for (var i=0;i<table_class_s.length;i=i+1){
30.             var tr_table_s = table_class_s[i].getElementsByTagName('tr');
31.             tr_table_s[0].getElementsByTagName('th')[0].colSpan = 8;
32.             for (var r=0; r<tr_table_s.length; r=r+1){
33.                 if
34.                 (tr_table_s[r].getElementsByTagName('th')[0].innerHTML.indexOf("Giorno") != -1) {
35.                     var rowSpan =
   tr_table_s[r+2].getElementsByTagName('td')[0].rowSpan
36.                     tr_table_s[r].innerHTML += "<th width=''"
   class='detail_table' valign='top' rowspan='2' colspan='1'>Calendario</th>";
```

```

36.         tr_table_s[r+2].innerHTML += "<td width='
    id='calendar"+i+"' class='detail_table' valign='center' rowspan='" + rowSpan + "'
    style='text-align:center;'></td>";
37.         createInput(i, r);
38.         break;
39.     }
40. }
41. }
42. } else if (document.URL.indexOf(docentPage) != -1) {
43.     var table_class_d = document.querySelectorAll('table.detail_table');
44.     for (var j=0;j<table_class_d.length;j=j+1){
45.         var tr_table_d = table_class_d[j].getElementsByName('tr');
46.         tr_table_d[0].getElementsByName('th')[5].colSpan = 4;
47.         tr_table_d[0].getElementsByName('th')[5].width = 80;
48.         for (var k=1;j<tr_table_d.length;k=k+1){
49.             tr_table_d[k].innerHTML += "<td width='
    id='calendar'+j+'_'+k+"' class='detail_table_middle' valign='center' style='text-
    align:center;' rowspan='1' colspan='1'></td>";
50.             createInput(j+"_"+k);
51.         }
52.     }
53. }
54. };
55.
56. stringCapitalize = function(word){
57.     var split_word = word.split(" ");
58.     var output = "";
59.     for (var j=0; j < split_word.length; j++){
60.         output = output + split_word[j].substring(0,1).toUpperCase() +
        split_word[j].substring(1,split_word[j].length).toLowerCase();
61.         output = output + " ";
62.     }
63.     return output;
64. };
65.
66.
67. getInfo = function(index, r){
68.     if (document.URL.indexOf(studentPage) != -1) {
69.         return getInfoStudent(index, r);
70.     } else if (document.URL.indexOf(docentPage) != -1) {
71.         return getInfoDocent(index);
72.     }
73. };
74.
75. getInfoDocent = function(index){
76.     //split index in table index and row index
77.     var table = index.charAt(0);
78.     var row = index.charAt(2);
79.     //retrieve all info-table
80.     var info_class = document.querySelectorAll('table.detail_table');
81.     //select the table specified by the table index
82.     var infotab = info_class[table].getElementsByName('tr');
83.     //retrive the html div with exam name and location
84.     var div_name =
    document.querySelectorAll('div#esse3old')[table].getElementsByName('table')[2]
    .getElementsByName('table')[0];
85.     //exam name
86.     var name_prefix =
    div_name.getElementsByClassName('legenda3')[0].textContent;

```

```

87.         name_prefix = name_prefix.substring(12,name_prefix.length);
88.         name_prefix = name_prefix.split(' ')[0];
89.         //exam type
90.         var name_suffix =
infotab[row].getElementsByTagName('td')[0].textContent;
91.         //name = exam name + exam type
92.         var type = examType(name_suffix);
93.         var name = type+" "+stringCapitalize(name_prefix);
94.         //retrieve date and hour and split them
95.         var date_hour = infotab[row].getElementsByTagName('td')[2].textContent;
// get "Data ora aula" text
96.         var date = date_hour.substr(0,10);
97.         var hour = date_hour.substr(11,5); // from time to end, if exist
98.         var place = date_hour.substr(17); // add start time
99.         //place = place.trim();
100.        hour = hour.replace(" ", "");
101.        //retrieve location
102.        name = name.replace("+","%2B");
103.        var description = "";
104.        info = ["","","","",""];
105.        //construct output
106.        info[0] = stringCapitalize(name);
107.        info[1] = date;
108.        info[2] = hour;
109.        info[3] = place;
110.
111.        var href =
infotab[row].getElementsByTagName('a')[0].getAttribute("href")
112.        $.ajax({
113.            type: "GET",
114.            url: href,
115.            async: false,
116.            success: function(htmlContent) {
117.                description =
$(htmlContent).find("textarea.tplForm").html();
118.                //alert(description);
119.                //console.log("ppp: "+descriprion)
120.            }
121.        });
122.
123.        info[4] = description;
124.        return info;
125.    };
126.
127.
128.    getInfoStudent = function(index, r){
129.        //retrieve exam name from hidden field
130.        var name = document.getElementsByName("AD_DES")[index].value;
131.
132.        name = name.replace("+","%2B");
133.        //retrieve all informations table
134.        var info_class =
document.querySelectorAll('table.detail_table');
135.        //select the rows of specified table (index)
136.        var infotab = info_class[index].getElementsByTagName('tr');
137.        var type = examType(infotab[r-
1].getElementsByTagName('th')[0].textContent);

```

```

138.             //retrieve date, hour and location
139.             var date =
infotab[r+2].getElementsByTagName('td')[0].textContent;
140.             var hour =
infotab[r+2].getElementsByTagName('td')[1].textContent;
141.             var place =
infotab[r+2].getElementsByTagName('td')[2].textContent; // add place for student
142.             var aula =
infotab[r+2].getElementsByTagName('td')[3].textContent;
143.             var description = "";
144.             //construct informations array
145.             info = ["","","","",""];
146.             info[0] = stringCapitalize(type+" "+name);
147.             info[1] = date;
148.             info[2] = hour;
149.             info[3] = place + " " + aula;
150.
151.             var href = $('a:first').attr('href');
152.             $.ajax({
153.                 type: "GET",
154.                 url: href,
155.                 async: false,
156.                 success: function(htmlContent) {
157.                     description =
$(htmlContent).find("div.titolopagina").html();
158.                     //alert(description);
159.                     //console.log("ppp: "+descriprion)
160.                 }
161.             });
162.
163.             info[4] = description;
164.             return info;
165.         };
166.
167.
168.         addToGoogleCalendar = function(info){
169.             //format informations
170.             var split_data = info[1].split("/"); // split date by char
171.             var split_ora = info[2].split(":"); // split time by : char
172.             var data = split_data[2] + split_data[1] + split_data[0]; //
data conversion in Google format (yyyymmdd)
173.             var oraInizio = split_ora[0] + split_ora[1]; // extract start
time
174.             //add 2 hour for all exams by default
175.             var duration = 2; // duration of exam
176.             var oraF = parseInt(split_ora[0])+duration; // set exam time
177.             var oraFine= oraF + split_ora[1]; // add exam time
178.             var text = info[0].replace(" ", "+");
179.             var where = info[3].replace(" ", "+");
180.             var description = info[4];
181.             //construct Google Calendar URL
182.             var indirizzo=
"https://www.google.com/calendar/render?action=TEMPLATE&src="+calendar_name+"&text="+text+
183.                 "&dates="+data+"T"+oraInizio + "00/"+data+"T"+oraFine
+"00&location="+where+"&details="+description+"%0A%0A&sf=true&output=xml";
184.             //Open new browser window due to confirm event adding.
185.             window.open(indirizzo, "blank");
186.         };
187.

```

```

188. examType = function(str){
189.     var scritto = "scritto";
190.     var laboratorio = "laboratorio";
191.     var orale = "orale";
192.     var scritta = "scritta";
193.     var empty = "";
194.     str = str.toLowerCase();
195.     if (str.indexOf(scritto) != -1)
196.         return stringCapitalize(scritto)+"-";
197.     if (str.indexOf(scritta) != -1)
198.         return stringCapitalize(scritta)+"-";
199.     if (str.indexOf(laboratorio) != -1)
200.         return stringCapitalize(laboratorio)+"-";
201.     if (str.indexOf(orale) != -1)
202.         return stringCapitalize(orale)+"-";
203.     return empty;
204. };
205.
206. window.addEventListener("load", main(), false);

```

COMMENTO DEL CODICE JAVASCRIPT DI S3-CALENDAR

Si fornisce una descrizione sommaria delle singole funzioni realizzate, tuttavia sarà possibile consultare il codice sorgente, opportunamente commentato, allegato a questo documento.

- Function main:** è il metodo principale dell'estensione. Per prima cosa verifica se si sta eseguendo il plug-in sulla pagina di "Esse3" riservata al docente, oppure su quella riservata allo studente. Questo controllo viene implementato in quanto le due pagine hanno un codice HTML differente. Dopo questo controllo, la funzione cerca nel DOM gli elementi di classe "detail_table". In questi elementi sono presenti le tabelle con all'interno tutte le informazioni relative agli appelli. Questa classe è presente sia sulla pagina del docente che su quella dello studente. Nel caso l'estensione venga eseguita sulla pagina dello studente, la funzione cercherà la riga in cui far comparire il tasto di aggiunta al calendario, sulla base delle intestazioni di colonna delle righe precedenti.

PROGETTAZIONE E PRODUZIONI DI CONTENUTI DIGITALI - [061670] - TERZO APPELLO 2013-2014								
Numero Iscrizione: 4 su 7								
Tipo Prova: orale								
Giorno	Ora	Edificio	Aula	Riservato per	Docenti Cognome Nome	Cancella	Stampa	Calendario
11/02/2014	10:00	Dipartimento di Informatica (Campus)		Nessun partizionamento	GENTILE ENRICHETTA			

SISTEMI PER LA COLLABORAZIONE IN RETE - [008051] - appello febbraio 2014								
Numero Iscrizione: 4 su 6								
Giorno	Ora	Edificio	Aula	Riservato per	Docenti Cognome Nome	Cancella	Stampa	Calendario
17/02/2014	17:00	c/o Studio del Docente		Nessun partizionamento	LANUBILE FILIPPO			

Per la pagine del docente invece, una volta individuata la singola tabella, il tasto sarà inserito in ogni riga escluso quella con le intestazioni di colonna.

Descrizione Appello	Data ora aula	Studenti iscritti	Esiti inseriti	Verbali caricati	Azioni
Prova orale e verbalizzazione	04/02/2015 11:00	1			
Prova di laboratorio C a distanza	03/02/2014 10:00	1			
Prova orale e verbalizzazione	20/11/2013 11:00				
Prova di laboratorio C a distanza	18/11/2013 09:00				
Prova orale e verbalizzazione	25/09/2013 11:00	2			
Prova di laboratorio C a distanza	23/09/2013 09:00	2			
Prova orale e verbalizzazione	09/09/2013 11:00				

In particolare il metodo non inserisce l'icona del calendario su cui cliccare, ma si limita a riformattare la tabella informativa, aggiungendo una nuova colonna in cui, tramite il metodo "createInput", sarà inserito il tasto a forma di calendario.

- **Function createInput:** questo metodo riceve in input due interi, uno che indica la tabella e l'altro che indica la riga in cui inserire il calendario. In particolare il numero di tabella è necessario in quanto nella stessa pagina, possono esserci più tabelle con le informazioni sugli appelli. Una volta inserito il tasto per il calendario, la funzione attiva un event listener, su di esso in modo che cliccandovi si attivi il metodo "doAction".
- **Function doAction:** riceve in input il numero di tabella, e di riga dalla quale carpire le informazioni da inserire nel calendario di Google. Chiama di seguito i metodi "getInfo" e "addToGoogleCalendar".
- **Function getInfo:** questo metodo riceve in input il numero di tabella e il numero di riga e restituisce in output un array con tutte le informazioni sull'appello selezionato. In particolare si avvale dei metodi "getInfoStudent" ed "getInfoDocent" a seconda che ci si trovi rispettivamente sulla pagina riservata allo studente o su quella riservata al docente.
- **Function getInfoStudent:** il metodo prende in input il numero di tabella e di relativa riga, quindi estrae dal DOM le informazioni sul nome dell'esame, tipo di appello, luogo, ora e data. Inserisce queste informazioni in un array e lo restituisce in output. Il metodo agisce sulla pagina riservata allo studente.
- **Function getInfoDocent:** il metodo prende in input il numero di tabella e di relativa riga, quindi estrae dal DOM le informazioni sul nome dell'esame, tipo di appello, luogo, ora e data. Le informazioni di luogo, data ed ora vengono recuperate dalla colonna "Data ora aula", dividendo la stringa in tre parti. Inserisce queste informazioni in un array e lo restituisce in output. Il metodo agisce sulla pagina riservata al docente. Questo metodo inoltre recupera informazioni riguardanti le "Note" dal link dell'esame di riferimento, presente nella colonna "Dati Appello" corrispondente alla pagina dei dati

dell'appello. Questa operazione è effettuata mediante una funzione jQuery sincrona che effettua una richiesta sincrona.

Appelli di: **LINGUAGGI DI PROGRAMMAZIONE + LABORATORIO** [visualizza dettagli >>](#)
[005745]
INFORMATICA (D.M.270/04) - BRINDISI [7912] (L)...

Dati appello (per selezionare più sessioni premere contemporaneamente il tasto CTRL)

*Data appello:	29/06/2015 (gg/mm/aaaa)	ora:	10 ▼ : 00 ▼
*Verbalizzazione:	Appello On-line con Firma Digitale ▼		
Tipo esame:	<input checked="" type="radio"/> Scritto <input type="radio"/> Orale		
*Iscrizioni (dal- al):	12/06/2015 (gg/mm/aaaa)	25/06/2015 (gg/mm/aaaa)	
*Descrizione:	Prova scritta ⓘ		
Prenotabile da:	tutti ▼		
Note:	Prova si svolgerà a Bari		
Sessioni:	2014/2015 - SECONDO PERIODO [01/06/2015 - 30/04/2016] ▲		

- **Function addToGooleCalendar:** il metodo riceve in input l'array con le informazioni sul singolo appello. Il nome ed il tipo di esame vengono posti nella forma "tipo – nome", in modo che possano interagire con un'altra estensione, che inserisce gli stessi dati su Wordpress. Tutte le informazioni vengono riscritte in modo compatibile con gli standard di Google Calendar. Ad esempio data e ora diventano un'unica stringa nel seguente formato: "20140116T090000/20140116T120000". La stringa indica data e ora di inizio appello, e data ed ora di fine appello. Una volta formattate tutte le informazioni, si invia la richiesta di inserimento a Google Calendar tramite un URL adeguatamente formattata

```
https://www.google.com/calendar/render?action=TEMPLATE&src=undefine  
&text=Scritto+%20Data%20Mining%20&dates=20140116T090000/20140116T1  
20000&location=Dipartimento+di%20Informatica%20%20(Campus)&details=  
Appello%20per%20fuoricorso%20%0A%0A&sf=true&output=xml.
```

Si inserisce il titolo dell'evento (text), ora e data di inizio e fine evento (dates), il luogo (location), eventuale descrizione da inserire nell'evento (details), ed il calendario in cui inserire l'evento (src). In particolare se Google Calendar non riceve alcun valore del calendario preferito, imposterà quello di default per l'utente. Il calendario preferito può essere specificato dal pannello di opzioni dell'estensione, inserendo l'identificatore univoco fornito da Google. Un esempio di identificatore univoco è il seguente:

abcde789caezt989@group.calendar.google.com

- **Function stringCapitalize:** questo è un metodo ausiliario, di formattazione delle informazioni. Il metodo prende in input una stringa composta da una o più parole e ne restituisce un'altra in cui la prima lettera di ogni parola è maiuscola e le restanti minuscole.
- **Function examType:** la funzione prende in input una stringa contenente il nome ed il tipo dell'esame. Da questa stringa estrae il tipo di esame tramite un'operazione di string matching. Questa funzione distingue in esame: "scritto", "laboratorio", "orale" e "prova scritta".

SUGGERIMENTI PER GLI SVILUPPATORI

Nelle ricerche effettuate per lo sviluppo dell'estensione "S3-Calendar", è stato trovato uno strumento molto utile per lo sviluppo, che offre numerosi vantaggi, ma anche un grosso svantaggio. Lo strumento di cui sopra, è una piattaforma online per lo sviluppo di estensioni, utilizzabile all'URL <http://crossrider.com>. Questa piattaforma permette di sviluppare delle estensioni per qualsiasi browser, inserendo solamente il codice Javascript da iniettare nelle pagine web. Crossrider crea automaticamente i pacchetti delle diverse estensioni, compatibili con Firefox, Chrome, Safari, Opera ed Internet Explorer. Tutte le proprietà dei plugin, come icone, menu e stili, sono configurabili tramite GUI (Graphical User Interface).



Se questo per gli sviluppatori può sembrare un sogno, in realtà non lo è affatto. L'estensione resterà sempre caricata sulla piattaforma e se si analizzano i sorgenti creati, al loro interno troviamo delle API che collegano l'estensione realizzata alla piattaforma in modo permanente. Questo significa che ogni qualvolta si modifica qualcosa su Crossrider, l'estensione viene aggiornata su tutti i browser sul quale è installata. Questo aspetto, che può sembrare vantaggioso, invece è una grossa falla nella sicurezza sia dell'estensione, sia del browser. Infatti basta distribuire una semplice estensione creata in questo modo ed in seguito modificare gli script al suo interno inserendo codice malevolo, e la sicurezza dell'utente finale è compromessa.

In conclusione quindi, questa piattaforma di sviluppo, può essere un ottimo punto di partenza, in quanto consente di testare la funzionalità degli script, svincolandosi dagli errori dovuti alla strutturazione non corretta delle estensioni. Questa infatti è la più grossa difficoltà quando ci si avvicina a questo tipo di sviluppo.

Spesso si scrive del codice Javascript perfetto, ma non si riesce a creare la struttura logica dell'estensione. Tramite Crossrider si possono testare immediatamente le funzionalità del plug-in in via di sviluppo, ma si consiglia di abbandonare la piattaforma una volta realizzato del codice funzionante, e passare alla strutturazione manuale del plug-in.

Nello sviluppo di "S3-Calendar" è stata seguita la strada consigliata, e si sono ottenuti ottimi risultati. Inoltre va specificato, che una volta strutturata l'estensione bisognerà apportare delle piccole modifiche ai codici Javascript, in quanto per determinate operazioni, i diversi browser, utilizzano direttive differenti e proprietarie, che vengono mascherate dalla piattaforma "Crossrider". Per concludere, questa piattaforma è utile ed allettante, solo per chi non ha mai sviluppato estensioni per un particolare browser, ma per un buon informatico, è consigliabile seguire la documentazione online, fornita dagli sviluppatori dei diversi browser, e guardare gli esempi e le guide disponibili in grandi quantità sul web.