La guida definitiva ai Chrome Dev Tool + i 13 migliori trucchi imparati con l'esperienza + come estendere Chrome con 5 add-on per il front-end

Parte 4 della collana I 10 strumenti essenziali se programmi con il front-end e Angular

Conosci gli strumenti **SEO di Chrome**?
Sai sfruttare al meglio i Chrome Dev Tools?
La tua pagina web implementa tutte le **best practice suggerite da Google**?
Imparerai tutto quello che c'è da sapere sui Chrome Dev Tools per sviluppare un progetto front-end, con gli strumenti nativi (Mobile, Elements, Network, Console, Application, Source, SEO e Best Practices) e estendendo Chrome con i migliori add-on per i Web Developer.

Autore

Ciao, sono Salvatore e trovi tutto su di me sul sito https://linkedin.com/in/salvatoreromeo. Ho scritto questo mini libro principalmente perché mi piace scrivere e condividere esperienza.

Nella vita mi occupo di sviluppo software fin da quando avevo 12 anni. Oggi è il mio lavoro principale. Insegno Ingegneria del Software all'Università degli Studi di Perugia e un corso di 3 giorni su Angular per le aziende insieme al Codemotion. Il corso è da poco stato trasformato in un libro che puoi trovare su Amazon.

Se vuoi maggiori informazioni su di me contattami pure tramite linkedin o tramite il sito https://devexp.io. Lì potrai anche trovare articoli sul front-end, altri libri del gruppo devexp.io e i codici sorgenti pubblicati su Github.

Sommario

La guida definitiva ai Chrome Dev Tool + i 13 migliori trucchi imparati con l'esperienza + come estendere Chrome con 5 add-on per il front-end

Parte 4 della collana I 10 strumenti essenziali se programmi con il front-end e Angular

Introduzione

Questo articolo chiude la serie di articoli dedicati ai migliori strumenti per velocizzare il processo di sviluppo front-end.
Abbiamo già visto Postman per verificare e gestire richieste HTTP e AutoHotkey per avere un assistente nello sviluppo front-end attraverso l'automazione di task noiosi e ripetitivi. Abbiamo inoltre visto i migliori strumenti per il deploy di un progetto front-end e cmder per rimpiazzare il prompt di Windows con un prompt moderno e ricco di funzionalità.

Chiudiamo la serie con i migliori strumenti di Chrome per gli sviluppatori. Per ogni strumento vedremo:

- le esigenze per cui utilizzarlo
- come sfruttarlo al meglio
- le tecniche e i trucchi più utili imparati con anni di esperienza

Vedremo inoltre come estendere Chorme con strumenti extra, per visualizzare oggetti JSON in maniera strutturata e per estendere le potenzialità di debug.

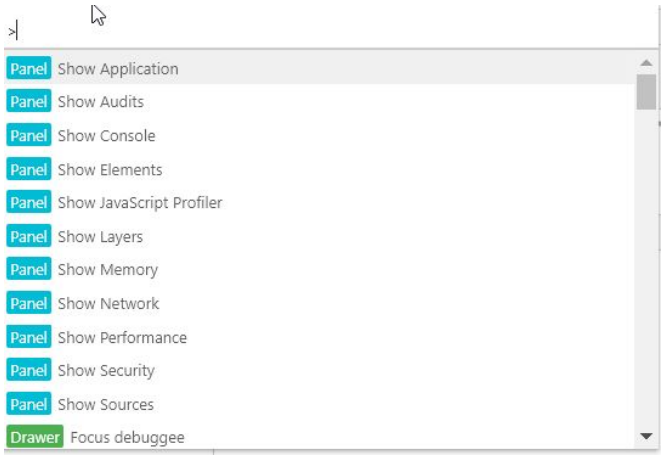
Chrome Dev Tools Hotkey

I Chrome Dev Tools sono supportati da Chrome senza installare nient'altro. L'unico requisito è quindi avere installato il browser Chrome. Anche di Chrome esiste una versione portable all'indirizzo: https://portableapps.com/apps/internet/google_chrome_portable

Per avviare i Chrome Dev Tools premiamo l'Hotkey F12:

Si aprirà un pannello con tutta una serie di Tab. Ogni Tab è uno strumento per il supporto allo sviluppo front-end che andremo a vedere di seguito con i migliori trucchi spesso meno noti per sfruttarlo al meglio.

Il primo e più importante Hotkey che dobbiamo imparare è CTRL+SHIFT+P:



Premendolo apparirà una finestra che ci elenca tutti i comandi possibili e possiamo cercarli anche con l'autocompletamento. Selezionando un comando, questo verrà eseguito.

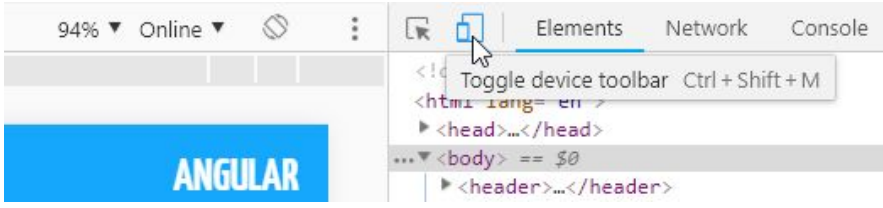
Chrome Dev Tools: dispositivi mobile + Tips

Esigenza

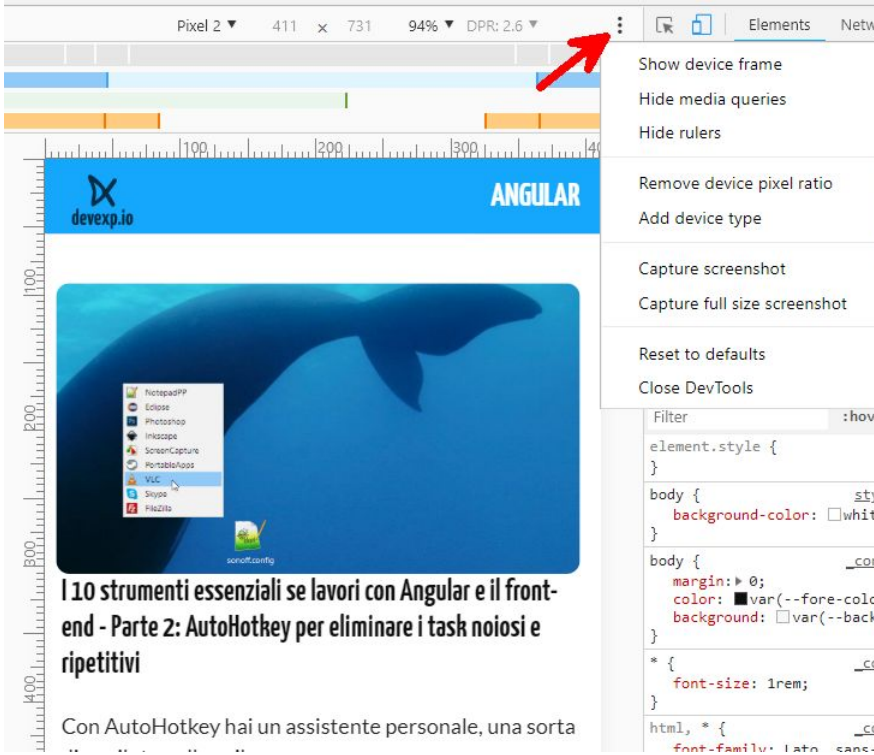
Vogliamo capire come si vedrà il nostro sito su dispositivi di varie dimensioni.
La cosa è molto utile soprattutto nel caso mobile. Chrome infatti permette di simulare un dispositivo mobile per darci un'idea di come si vedrà una pagina su quel dispositivo. Ricordiamo che si tratta di simulazione e che su un dispositivo vero potrebbe vedersi in modo leggermente diverso.

Gli strumenti per il Mobile

Per avviare lo strumento di simulazione mobile usiamo l'icona in alto a sinistra dei tab:



Per il mobile sono presenti diversi ulteriori strumenti accessibili con i tre puntini poco più a sinistra del pulsante appena cliccato. Cliccandoci apparirà un menu che mostra le varie funzioni:



Tra queste le più interessanti sono:

- La cattura di uno screenshot (area visibile o intera pagina)
- Il ruler, o righello, per capire velocemente la posizione e dimensione di un qualsiasi elemento
- Le barre delle media queries, che ci dicono se e dove abbiamo messo dei breakpoint nelle nostre media queries CSS

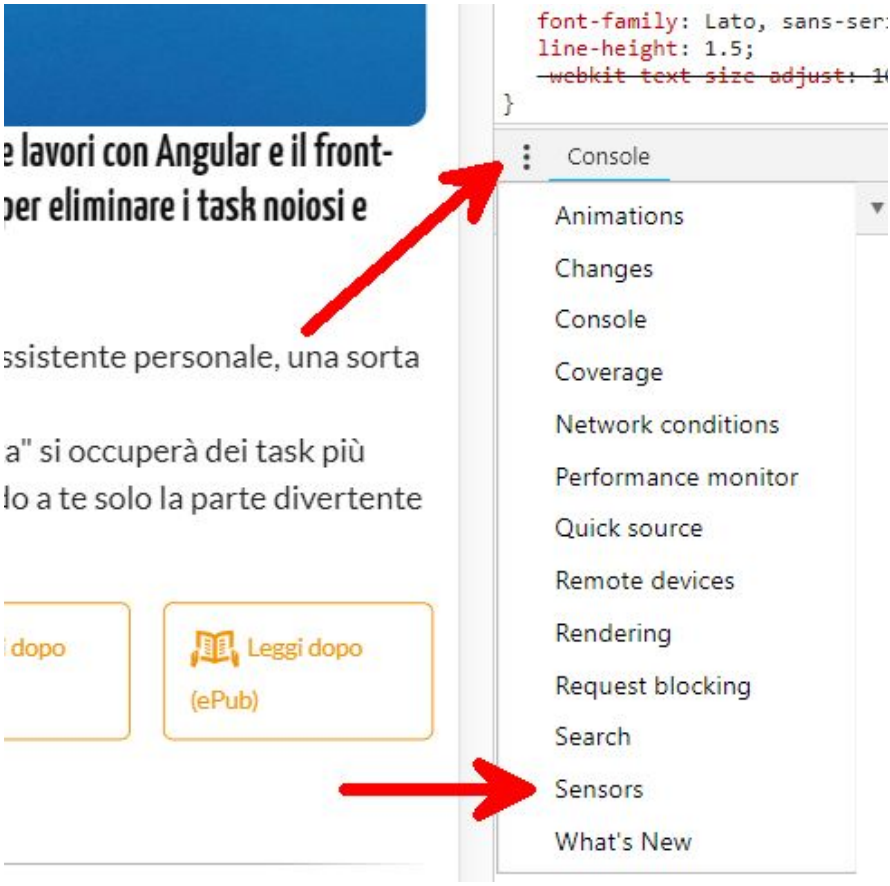
Con la barra superiore è possibile invece:

- Cambiare dispositivo

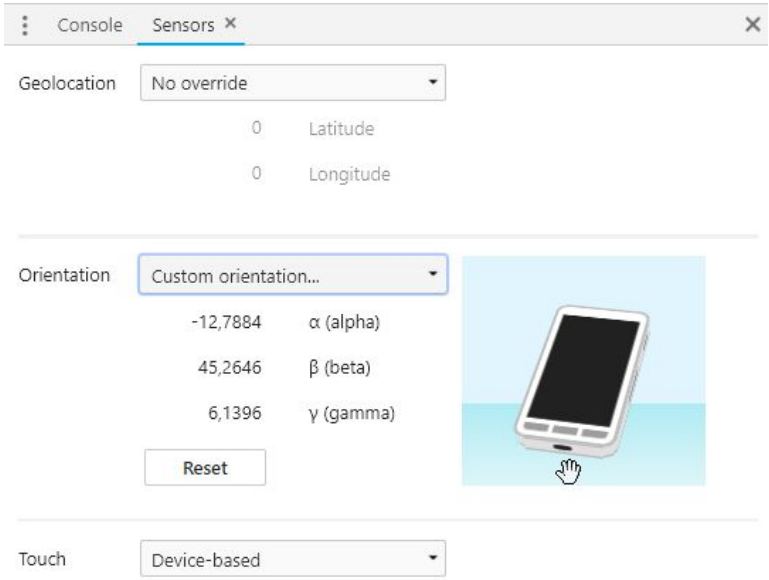
- Cambiare il livello di zoom
- Simulare una connessione più o meno veloce
- Cambiare l'orientamento da portrait a landscape

Tip #1: Gestione dei sensori

Sapevate che è possibile simulare quasi tutti i sensori di un dispositivo mobile tramite un pannello secondario?
Dalla *Console* in basso (sia apre premendo Esc se non è presente), premendo nei tre puntini a sinistra del Tab *Console*, è presente tutta una serie di ulteriori strumenti, tra cui i *Sensors*.

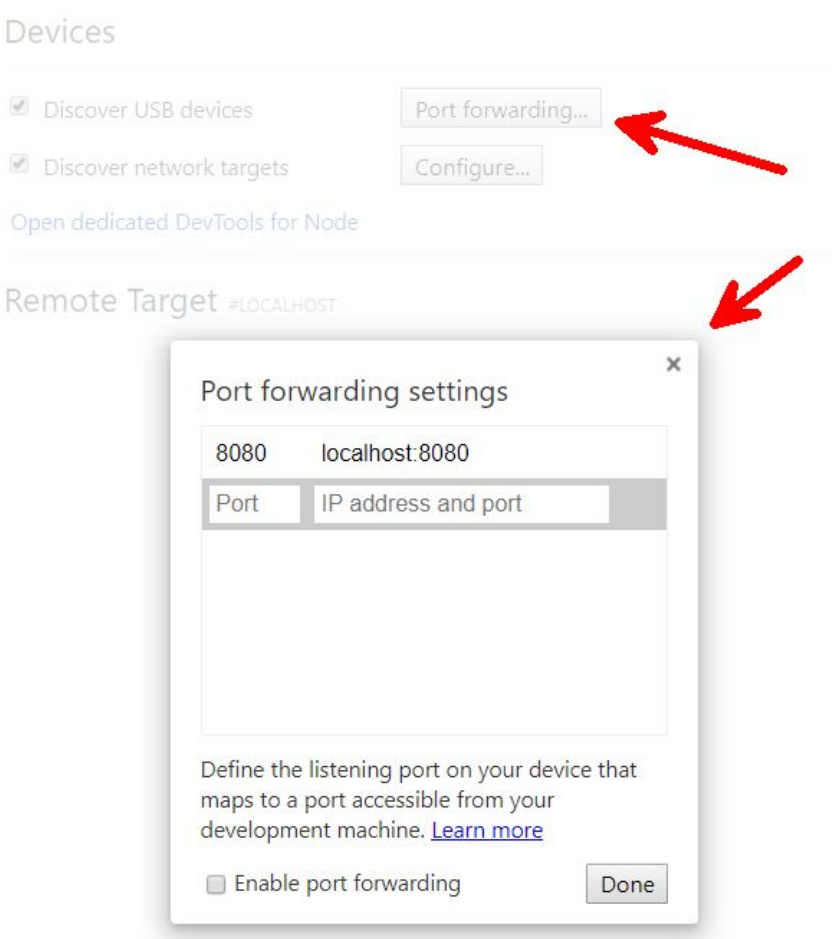


Cliccandoci si apre un pannello per gestire i sensori e simulare i valori di GPS (Geolocation), accelerazione (Orientation), e tipo di touch.



Tip #2: Strumenti per dispositivi mobile reali

Collegando un dispositivo mobile tramite USB al nostro PC e andando sul link chrome://inspect/#devices potremo vedere il nostro dispositivo tra i *Remote devices* e ispezionare un tab di Chrome con quasi tutti gli stessi strumenti disponibili su desktop.
Usando il pulsante *Port Forwarding* possiamo anche vedere sul dispositivo un sito web che gira su un server in *localhost* sul nostro PC:


Devices
Discover USB devices
Port forwarding...
Discover network targets
Configure...
Open dedicated DevTools for Node
Remote Target #LOCALHOST
Port forwarding settings
8080
localhost:8080
Port
IP address and port
Define the listening port on your device that maps to a port accessible from your development machine. Learn more
Enable port forwarding
Done

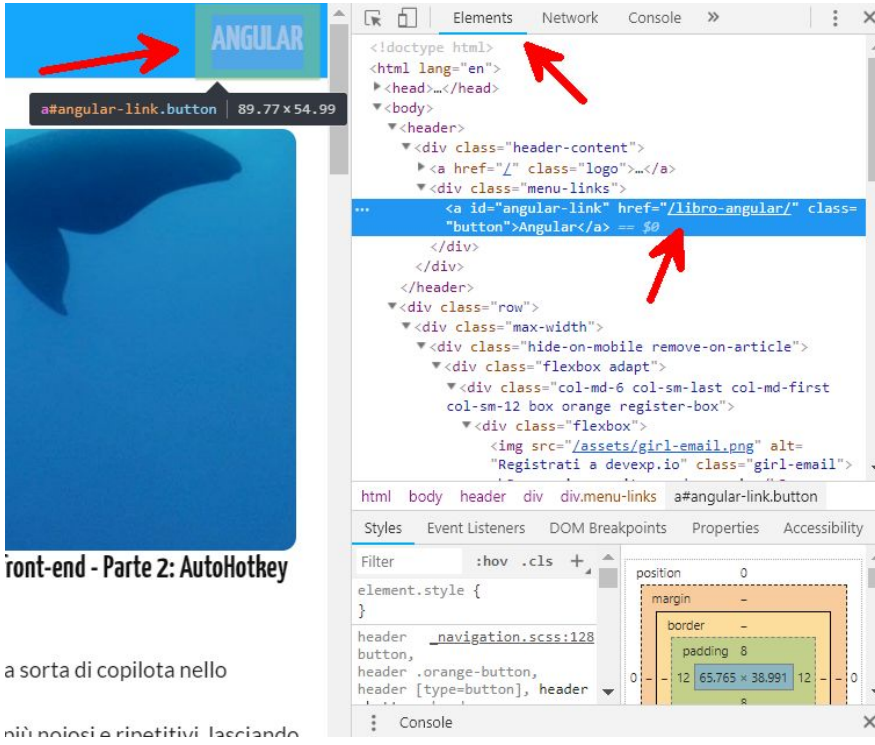
Chrome Dev Tools: Elements + Tips

Esigenza

Quando creiamo un qualunque progetto front-end, la pagina è progettata con il linguaggio HTML. Avremo speso la necessità di esplorare l'HTML e magari modificarlo live per vedere come verrebbe.

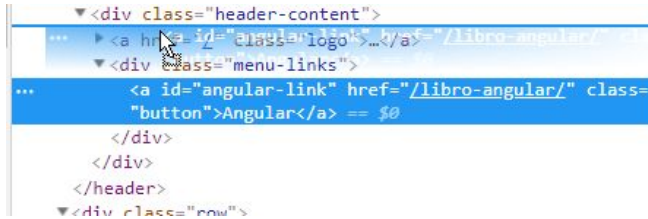
Lo strumento Elements

Per ispezionare un elemento HTML possiamo aprire il tab *Elements* e, posizionandoci su un qualsiasi tag HTML, si evidenzierà il corrispondente elemento nella pagina:

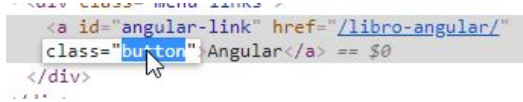


Possiamo interagire con i tag in vari modi:

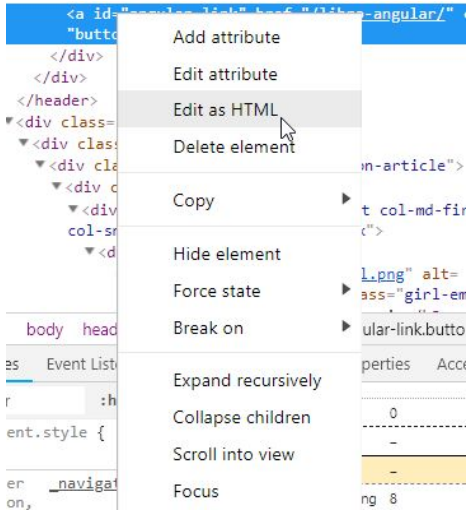
- eliminare un tag premendo *Canc*
- trascinare un tag in un altro tag



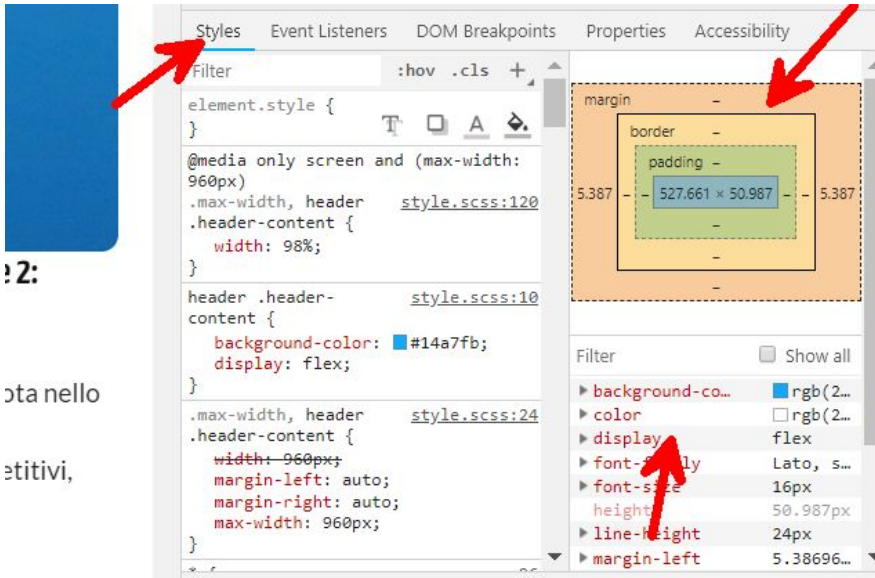
- modificare il contenuto di un attributo cliccando due volte sul suo valore



- Modificare il codice HTML o copiare l'elemento con il tasto destro

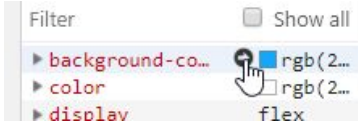


Il tab *Elements* contiene inoltre diversi strumenti per ispezionare la parte CSS della nostra pagina.



Con il sotto-pannello *Styles* possiamo valutare le regole CSS applicate all'elemento selezionato nel tab *Elements* e modificare live colori, dimensioni o in generale tutti i valori CSS.

Nella parte destra del pannello sono mostrate tutte le regole effettivamente applicate ad un elemento e cliccando su una regola è possibile capire in quale classe CSS è stata dichiarata:

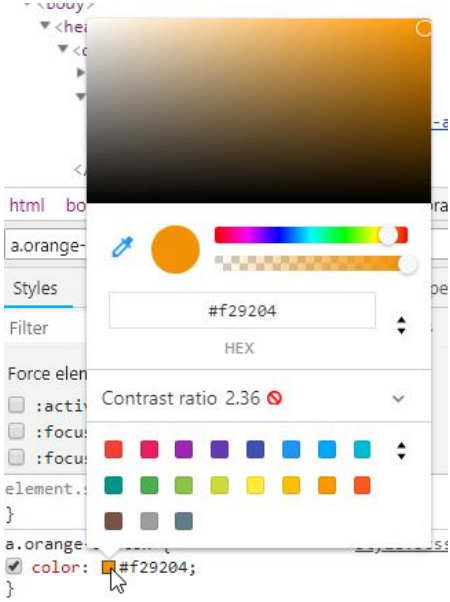


Altri strumenti interessanti meno noti sono

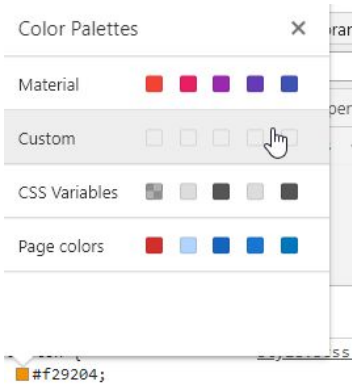
- Color picker
- Shadow editor
- Animation Editor

Color picker

Se una regola CSS è un colore, possiamo cliccare accanto all'icona quadratino colorato per aprire lo strumento.

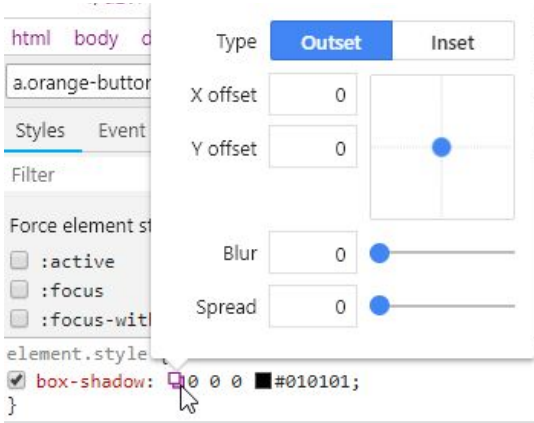


Oltre a cambiare colore, premendo l'icona a doppie frecce superiore possiamo cambiare il formato, mentre premendo l'icona a doppie frecce inferiore Chrome ci suggerisce delle palette predefinite, come la palette Material, oltre a mostrarci i colori usati nella pagina corrente:



Shadow editor

Lo shadow editor è attivabile in maniera analoga per le regole relative alle ombre:

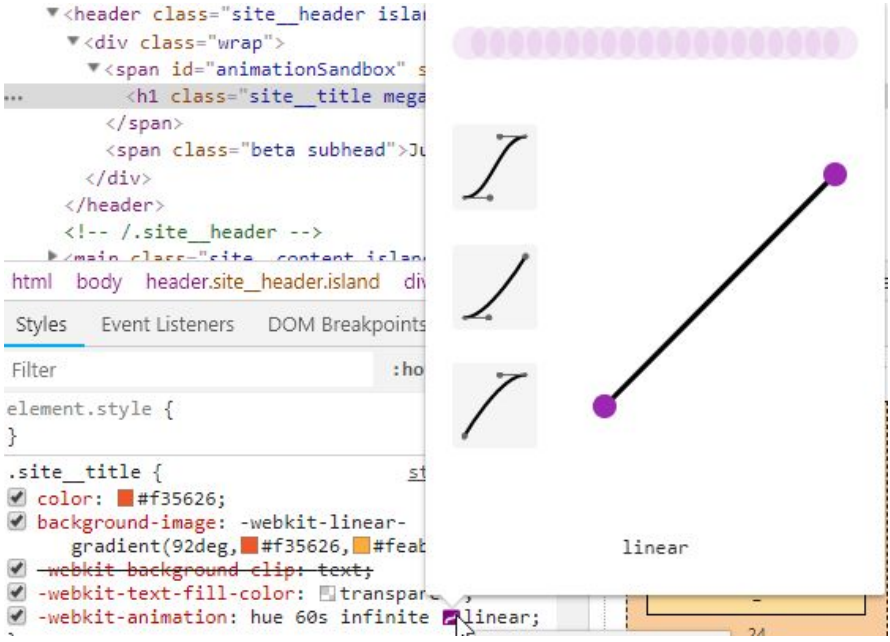


Se non ricordiamo la sintassi della regola CSS, da qui possiamo impostare tutti i parametri con una schermata visuale.

Animation editor

Quando vogliamo aggiungere al nostro sito delle animazioni per migliorare l'esperienza utente, possiamo fare riferimento al sito https://daneden.github.io/animate.css/ in cui è possibile

scaricare il file *animate.css* con tante animazioni semplici, pulite e non pesanti.
Anche alcune regole delle animazioni possono essere modificate in Chrome premendo sull'iconcina con la curva, nella regola relativa all'animazione:

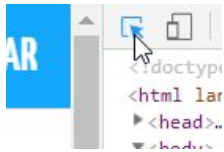


Dopo questa panoramica, vediamo alcuni trucchi per lavorare con gli elementi.

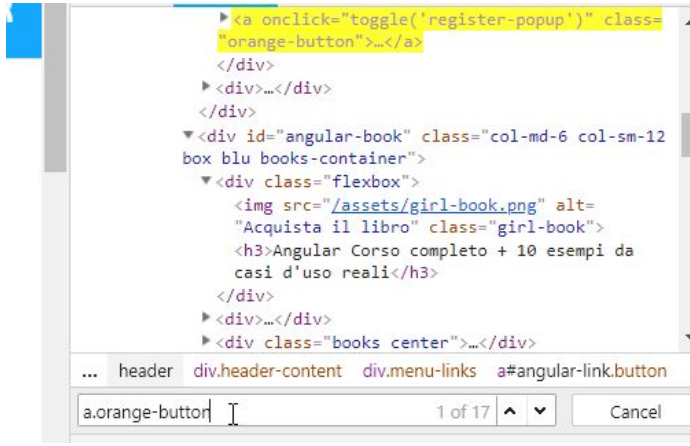
Tip #3: Ricerca rapida di un elemento

Per ricercare un elemento abbiamo diverse possibilità:

- In maniera visuale, usando lo strumento Inspector (accessibile anche con CTRL+SHIFT+I)



- Premendo CTRL+F nel tab *Elements* e ricercando per selettore, con una stringa contenuta nel testo dell'elemento o ancora usando Xpath



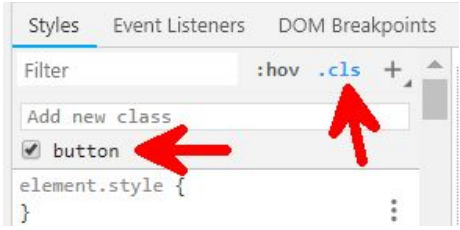
- Dalla *Console* usando l'operatore simil-jquery $ e $$:

(spostandoci con il mouse su un elemento della *Console* questo verrà anche evidenziato nella pagina; cliccandoci

sopra col tasto destro potremmo evidenziarlo anche nel pannello *Elements*)

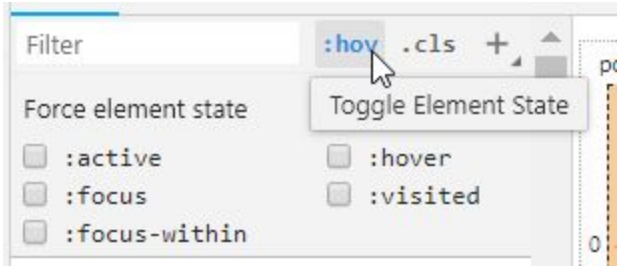
Tip #4: Abilitare/disabilitare le classi di un elemento

Per ogni elemento selezionato nel tab *Elements* possiamo fare il "toggle" delle classi usando il pulsante *.cls* nel pannello *Styles*:



Tip #5: valutare la grafica istantanea di un elemento

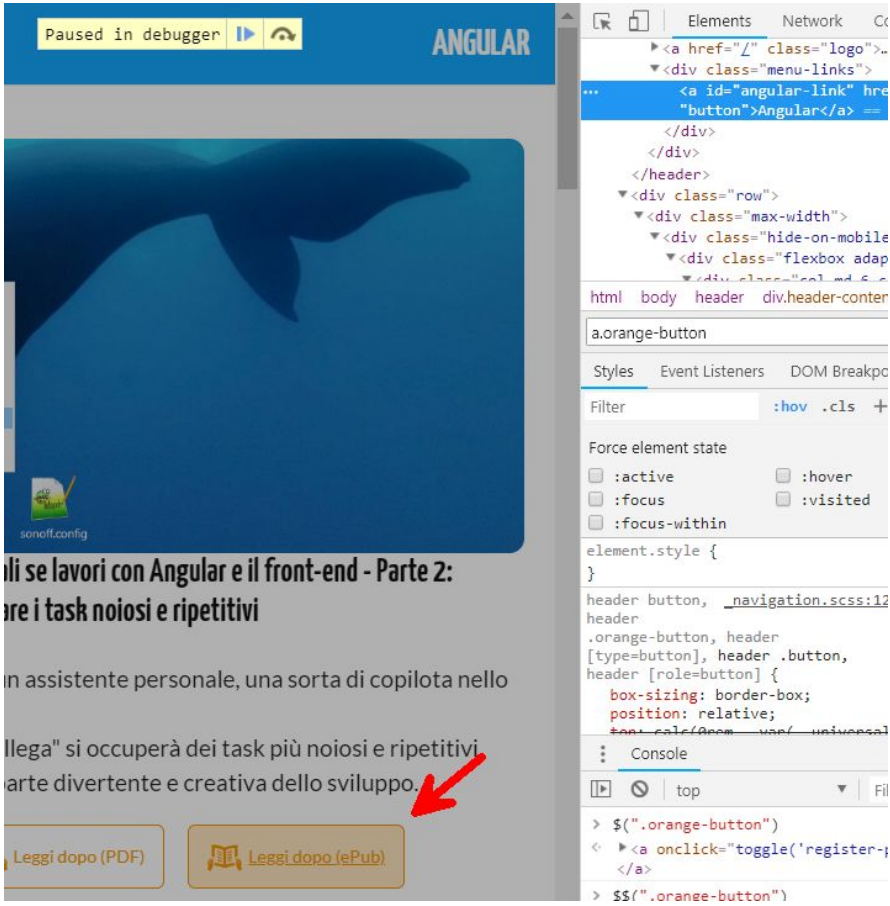
Chrome mette a disposizione un checkbox per bloccare lo stato di un elemento, ma non sempre è sufficiente in quanto spesso un elemento cambia grafica non in base a regole CSS, ma ad esempio in base a script JavaScript.
Lo stato può essere bloccato usando il pulsante *.hover* vicino al pulsante.*cls* precedente:



Ma quando lo stato non basta, per valutare in ogni caso la grafica istantanea di un elemento seguiamo questi passi:

1. Andiamo nel *Tab Source*

2. premiamo F8, che bloccherà la pagina
3. torniamo nel pannello *Elements* e valutiamo le varie regole



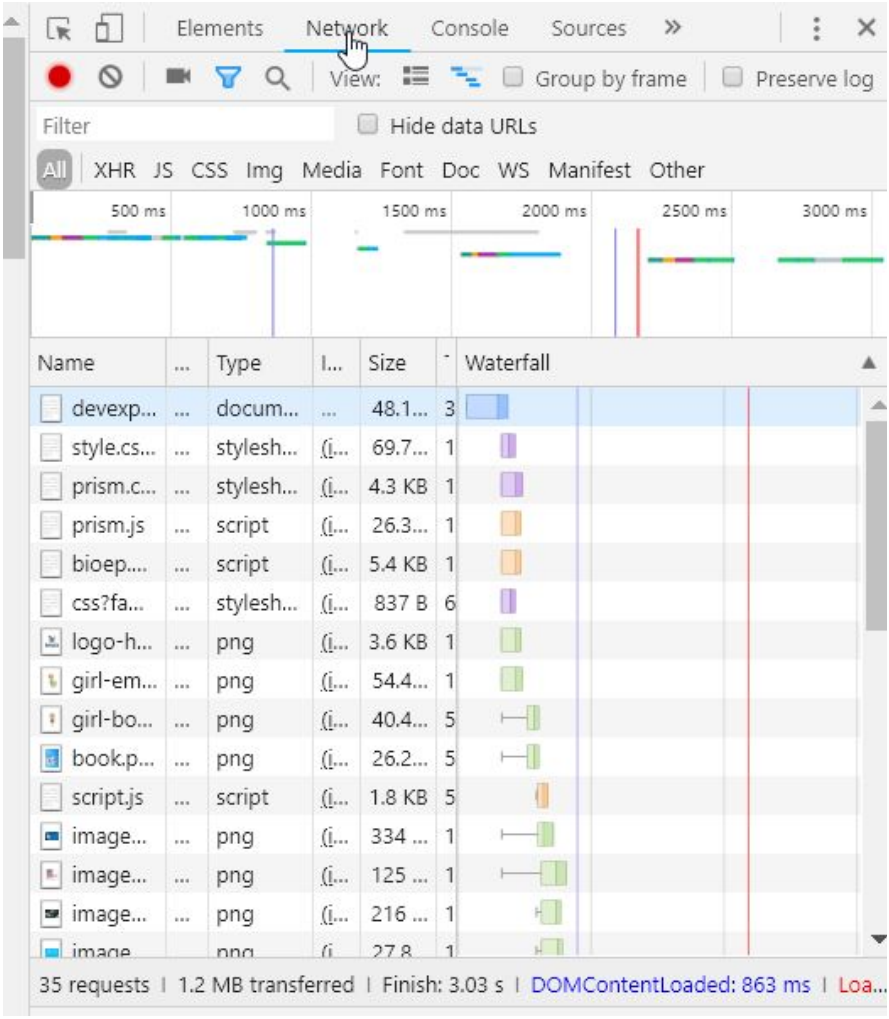
Chrome Dev Tools: Network + Tips

Esigenza

Un progetto front-end comunica con un back-end per inviare e richiedere informazioni. È utile sapere cosa inviamo e cosa riceviamo e in questo caso ci aiuta il pannello Network.

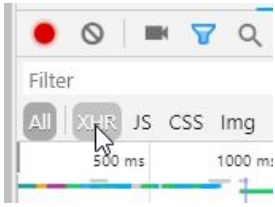
Gli strumenti Chrome per le richieste/risposte HTTP

Aprendo il pannello *Network* ci ritroveremo con la lista delle richieste al back-end.

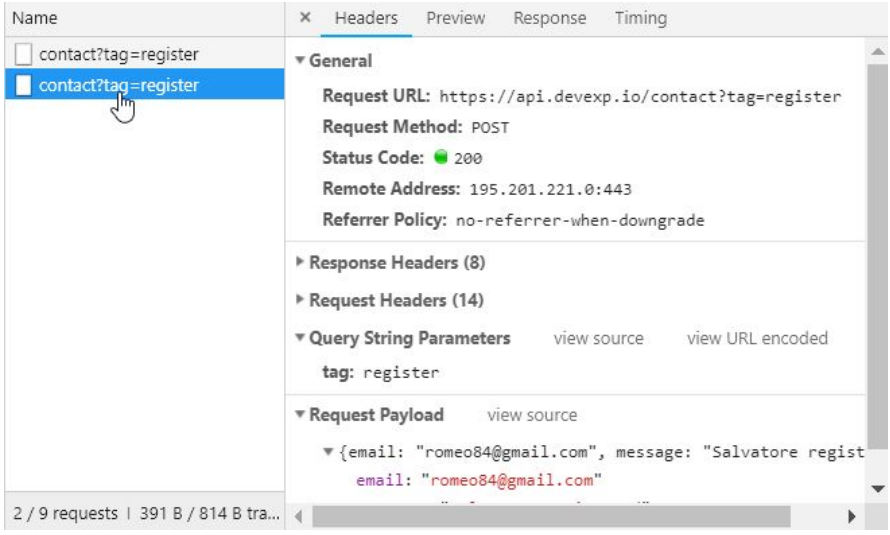


Trattandosi spesso di molte richieste, Chrome ha messo a disposizione molti strumenti per filtrare la lista. Possiamo specificare un filtro testuale nel campo *Filter* oppure possiamo

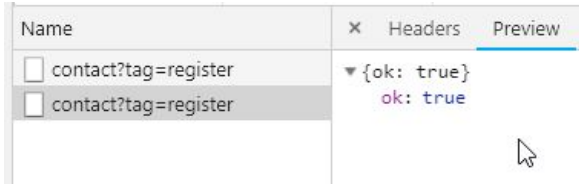
usare i filtri per tipo: Script, immagini, CSS, ..., ma il più usato per lo sviluppo front-end moderno è sicuramente quello per filtrare le richieste asincrone: *XHR*



Cliccando su una richiesta, si apre sulla destra un pannello con tutta una serie di sotto-pannelli contenenti informazioni utili.

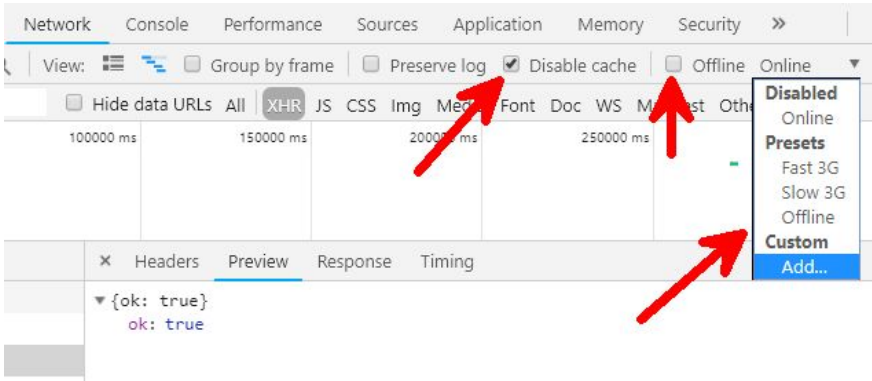


Molto interessanti le informazioni inviate al back-end, nella sezione *Request Payload* (utili soprattutto nel caso di una POST). Per esaminare la risposta del back-end possiamo invece cliccare sul Tab *Preview*



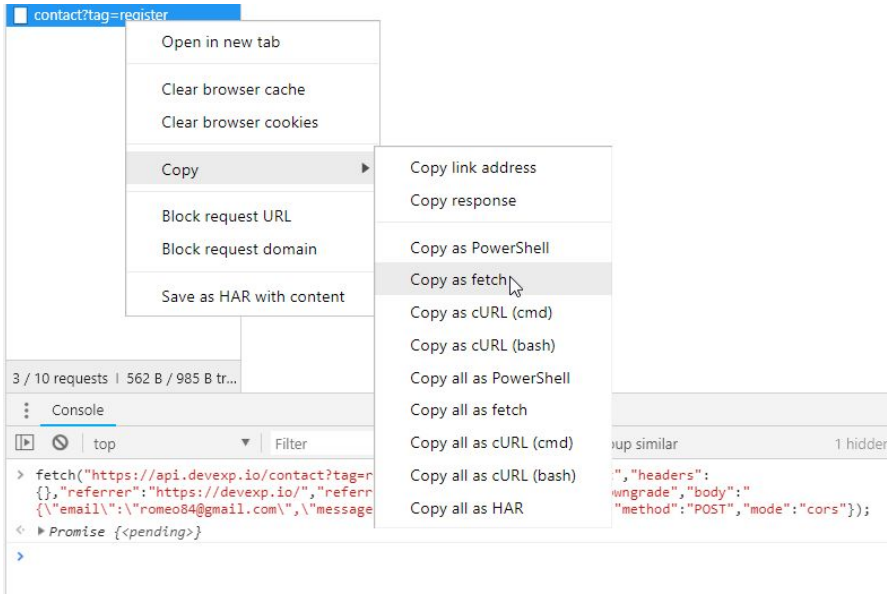
Lo strumento Network non è solo uno strumento per investigare le richieste HTTP, ci permette infatti di:

- disabilitare la cache del browser
- andare offline
- simulare in maniera più fine la velocità della connessione, usando il dropdown in alto a destra



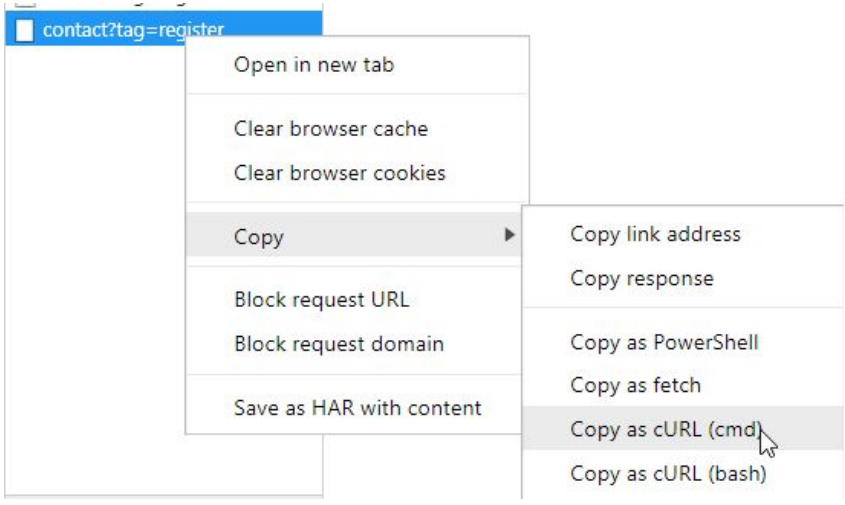
Tip #6: Ripetere una richiesta HTTP tramite console

Se clicchiamo col tasto destro su una richiesta abbiamo diversi strumenti. Copiando la richiesta come fetch (*Copy as Fetch*) possiamo ripeterla nella *Console* semplicemente incollandola e premendo *Invio* (e possiamo eventualmente modificarla prima di premere *Invio*):



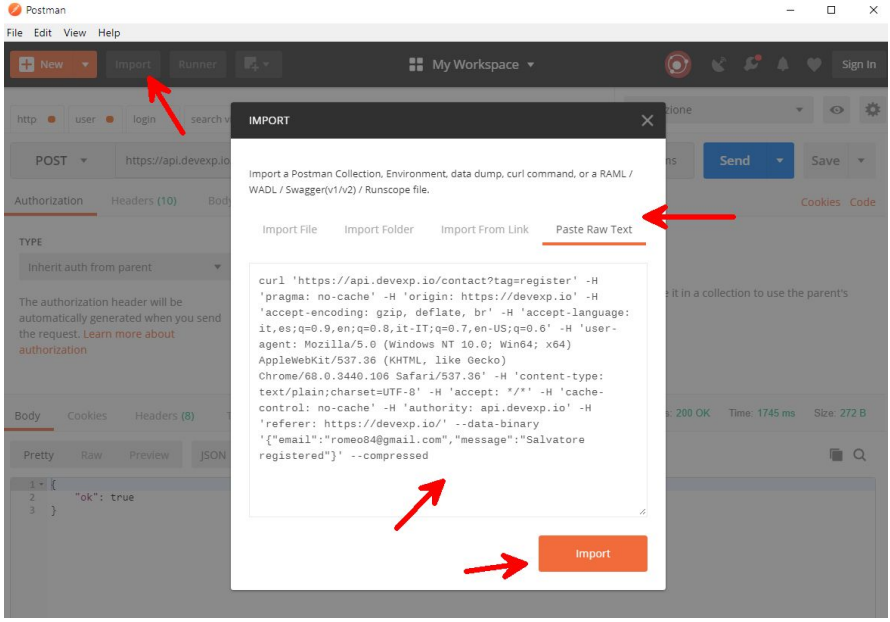
Tip #7: Copiare una richiesta in Postman o eseguirla con cUrl

Per importare una richiesta Chrome in Postman dobbiamo trasformarla in formato *cUrl*:



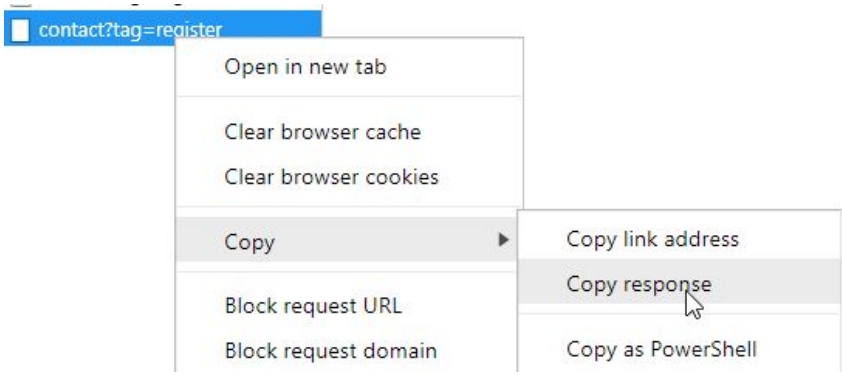
A questo punto possiamo importarla in Postman:

- Premiamo su *Import*
- Premiamo su *Paste Raw Text*
- Incolliamo il comando e premiamo *Import*

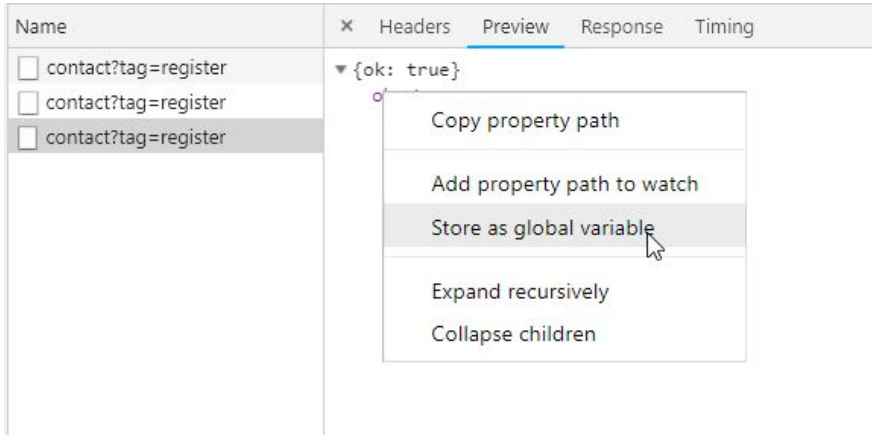


Tip #8: Copiare il JSON di risposta del server nella clipboard

Per copiare il JSON restituito dal server possiamo usare ancora una volta il menu contestuale



Oppure possiamo copiare parte del JSON in una variabile temporanea da manipolare nella *console*:

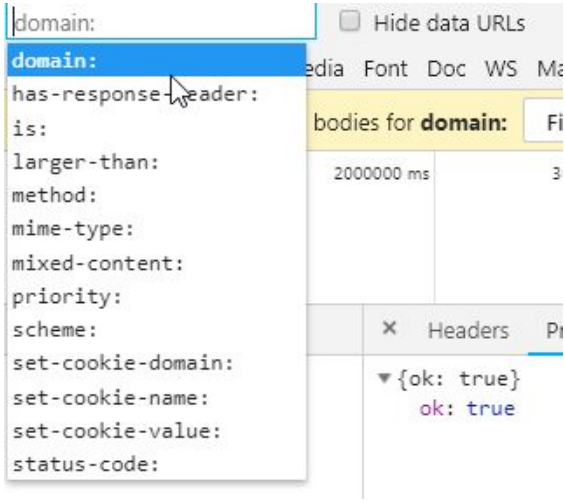


La variabile è ora disponibile con il nome *temp1*

Tip #9: Filtrare le richieste in maniera avanzata

Il campo *Filter* che abbiamo visto in precedenza per filtrare gli URL che contengono un particolare testo, può anche essere utilizzato per filtrare in maniera più avanzata. Premendo

CTRL+SPAZIO nel campo appare un menu che ci permette di filtrare le richieste con una caratteristica specifica:

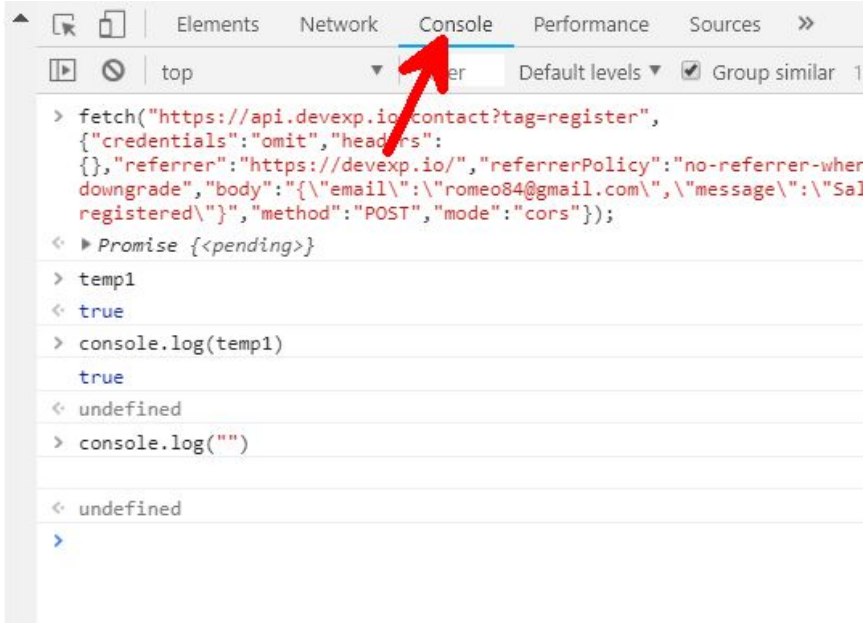


Ad esempio possiamo filtrare le risposte errate con *status-code: 4* o anche quelle con dimensione maggiore di 10 KB: *larger-than:10k*

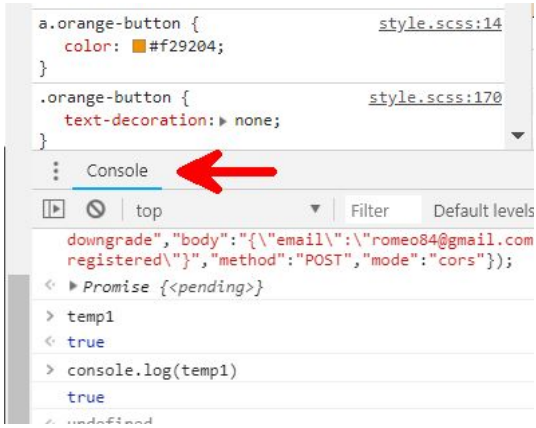
Chrome Dev Tools: Console + Tips

Esigenza

Quando scriviamo il codice in qualunque linguaggio, abbiamo la possibilità di stampare il risultato di un'espressione a video. La console ci mostra i risultati, ma non è esclusivamente uno strumento passivo: tramite la console possiamo interagire con la nostra applicazione ed è talmente utilizzata che Google ha previsto un doppio pannello: uno in alto a fianco agli altri Tab:



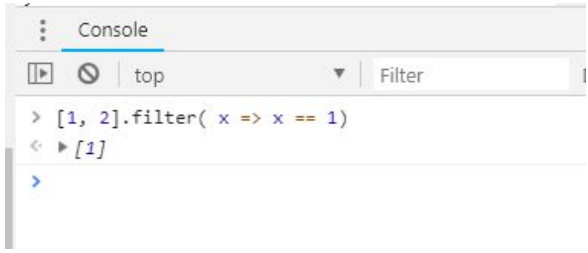
E l'altro, identico, sempre presente in concomitanza di un altro Tab premendo ESC:



La console di Chrome

Le principali funzioni della *console* sono:

- esecuzione di qualsiasi codice JavaScript ES6 sfruttando l'autocompletamento e l'accapo smart quando premiamo INVIO



- copia di un oggetto o stringa nella clipboard con la funzione *copy*

- Fare l'inspect di un elemento nel pannello *Element* con l'istruzione *inspect($("SELETTORE"))*

```
> inspect($("a.orange-button"))
<- ▶<a onclick="toggle('register-popup')" class="orange-button">…</a>
```

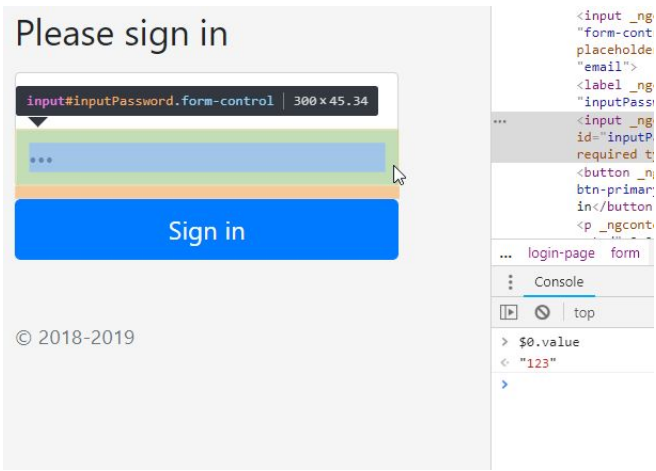
- Premiamo CTRL+L (o CTRL+K dipendentemente dal Sistema Operativo) per pulire la *console*

Possiamo inoltre usare alcune funzioni della *console* direttamente dal nostro codice JavaScript, ad esempio possiamo interrompere il flusso di codice scrivendo l'istruzione *debugger;* nel nostro file sorgente, subito prima del codice da debuggare.

Tip #10: visualizzare la password salvata in un input

Vi svelo questo piccolo trucco, ma usatelo solo per visualizzare le vostre password salvate in Chrome e dimenticate:

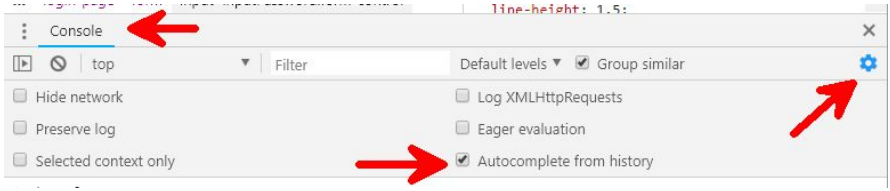
1. Ispezioniamo l'input password con CTRL+SHIFT+I e poi cliccandoci sopra
2. Scriviamo nella console *$0.value*



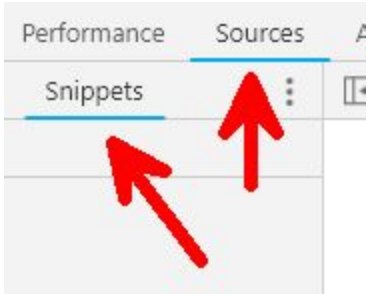
In generale $0 ci fornisce il riferimento all'ultimo elemento ispezionato, che in questo caso è l'input contenente la password. L'istruzione *$0.value* nella console ce ne mostra il valore.

Tip #11: usare le snippet dalla console

La *console* può mantenere la storia di tutti i comandi digitati e ricordarceli tramite l'autocompletamento. Basta andare nelle impostazioni della *console* stessa (dove ci sono anche altre configurazioni):



Se però vogliamo salvare snippet complesse di codice possiamo usare il pannello *Snippet* di Chrome, disponibile nel Tab *Source* (che vedremo tra poco) e poi cliccando sul sotto-pannello *Snippet*



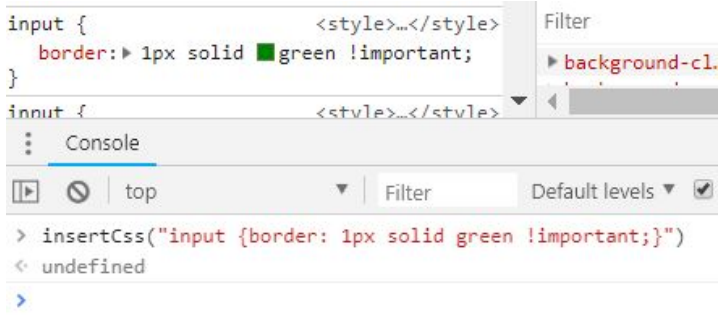
Qui possiamo crearne di nuove e salvarne il codice.

Possiamo poi richiamare queste snippet dalla *console:*

- premiamo CTRL+O
- digitiamo il nome della snippet
- premiamo CTRL+INVIO per eseguire la snippet in *console*

Possiamo salvare le nostre snippet preferite o aggiungere snippet create dalla community (copia/incolla del codice in una nuova snippet), come le seguenti:

- Insert CSS per inserire regole CSS inline



- ViewCookies per stampare i cookie
- Print Query String per stampare i parametri dell'URL

Chrome Dev Tools: Application + Tips

Esigenza

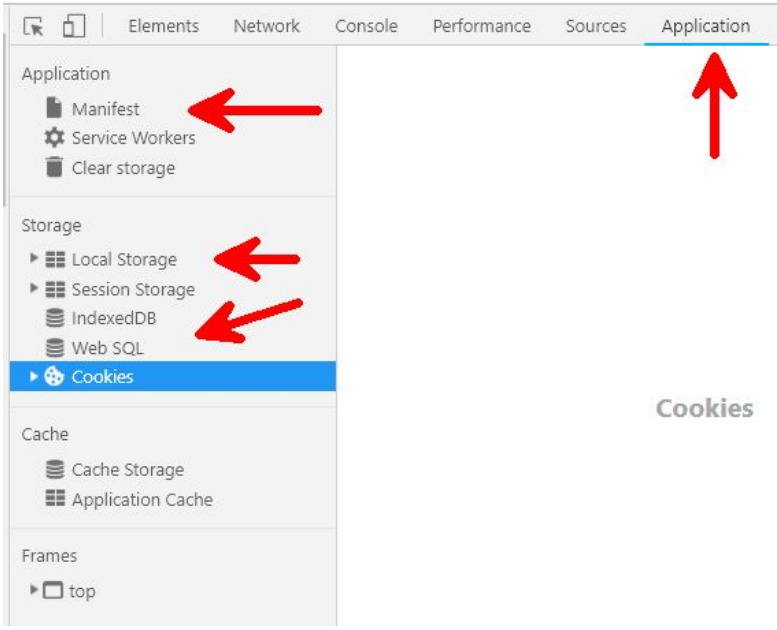
Un'applicazione web moderna può salvare molte informazioni in maniera persistente. Ciò significa che chiudendo il browser e riaprendolo le informazioni sono ancora a disposizione. Avremo spesso bisogno di esplorare queste informazioni e magari anche modificarle.

Gli strumenti Chrome per la persistenza

Possiamo salvare informazioni in un progetto web usando una di queste tecnologie:

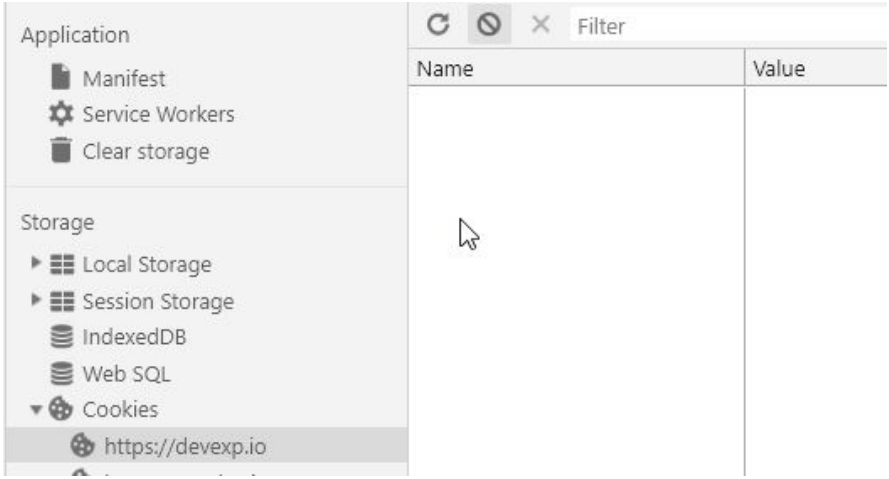
- cookie
- localstorage
- database SQL con WebSql
- database NoSql con IndexedDB
- File offline con AppCache o service Worker

Tutte queste informazioni le possiamo ispezionare dal pannello Application:



Gli strumenti che useremo maggiormente e che vedremo con qualche dettaglio in più sono la sezione *Cookie*, *LocalStorage*, *WebSQL* e *IndexedDB*.
La sezione *Manifest* permette di esplorare l'AppCache della pagina, che serve per far funzionare un sito web anche offline in assenza di connessione.
Oggi è sempre meno utilizzata ed è stata sostituita dai *Service Worker*.

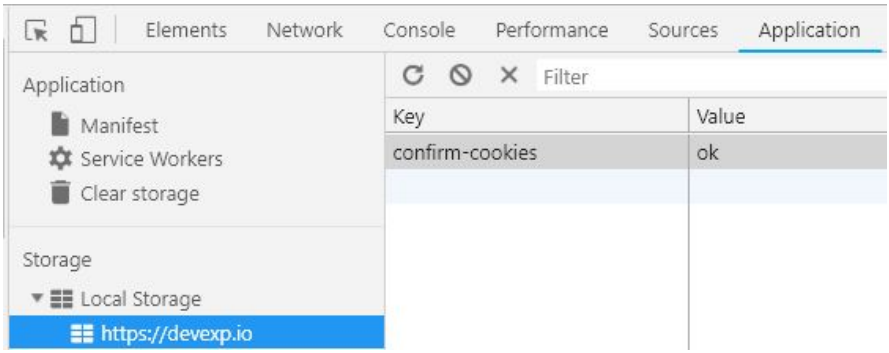
Cookie



In questa sezione possiamo esplorare i cookie di un sito web, modificarli ed eliminarli.

LocaleStorage

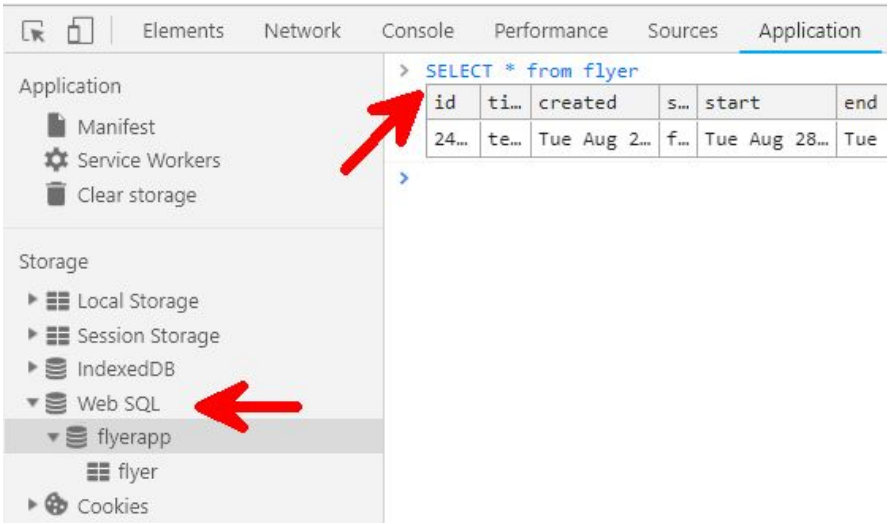
Sempre più spesso al posto dei cookies si utilizza il LocaleStorage per memorizzare piccole informazioni da inviare al server in maniera controllata dal client, Ad esempio possiamo qui memorizzare un token, da inviare con gli *headers* HTTP.



Come per il pannello dei cookies, anche qui possiamo modificare ed eliminare tutti i valori.

WebSQL

Dopo aver creato un DB sql con le API relative, possiamo esplorare il DB o eseguire script SQL per manipolare il DB usando il pannello:



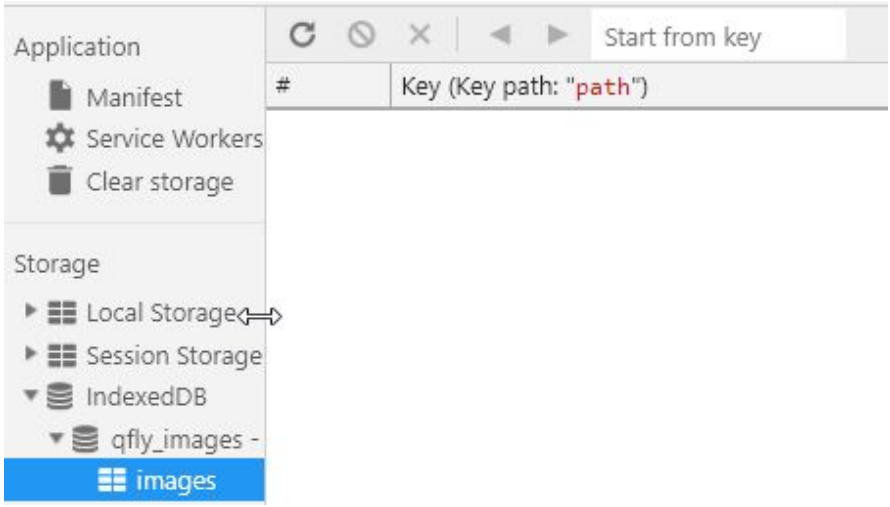
Tutte le operazioni SQL sono supportate:

- INSERT
- UPDATE
- SELECT
- DELETE
- CREATE

IndexedDB

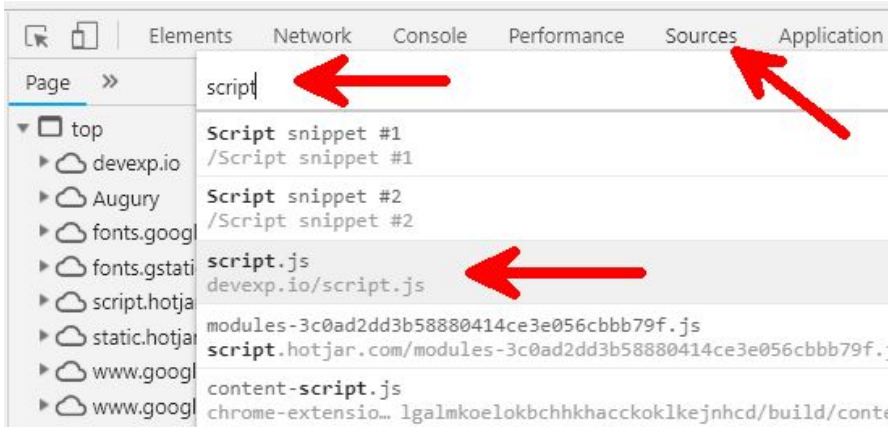
Esiste anche un DB NoSql nel browser, noto come IndexedDB. Per esplorarlo possiamo usare il pannello relativo che,

nonostante non sia potente come il precedente, permette di esplorare il DB ed eseguire query base:



Chrome Dev Tools: Sources + Tips

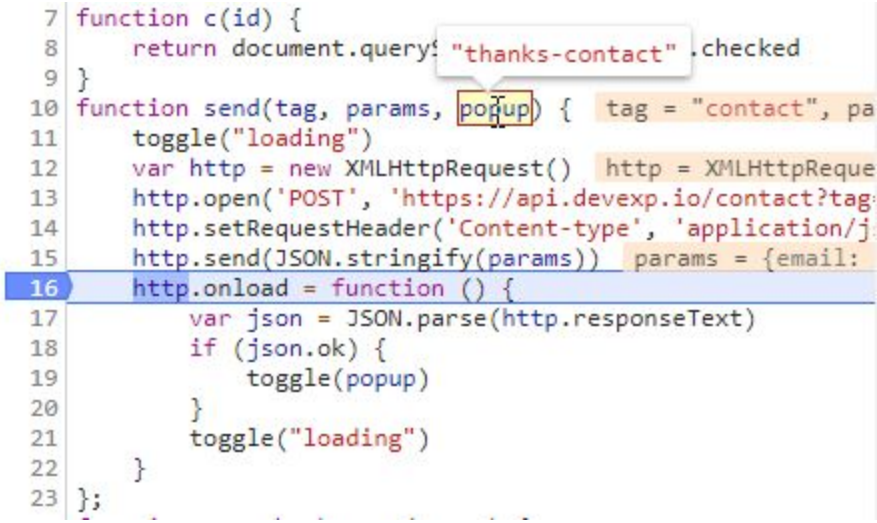
E arriviamo infine al pannello più potente per noi sviluppatori: *Source*.
Lavorando col codice, da qui avremo un pieno controllo di tutto quello che sta succedendo.
Per esplorare i codici della pagina possiamo usare la barra laterale, o, come preferisco, premere CTRL+P col pannello Source attivo e iniziare a scrivere il nome del file:



Dopo aver aperto uno script possiamo aggiungere un breakpoint per fermare il flusso di codice facendo doppio click sulla linea interessata:

```
function send(tag, params, popup) {
    toggle("loading")
    var http = new XMLHttpRequest()
    http.open('POST', 'https://api.
    http.setRequestHeader('Content-
    http.send(JSON.stringify(params
    http.onload = function () {
        var json = JSON.parse(http.
        if (json.ok) {
            toggle(popup)
        }
        toggle("loading")
    }
```

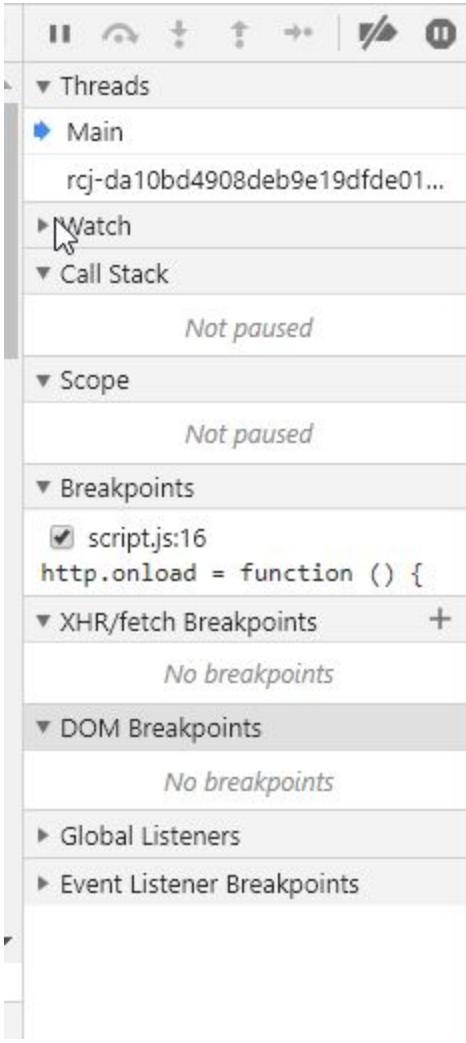
Quando il programma è bloccato, possiamo esplorare il contesto delle variabili passando il mouse sopra una variabile:



```
function c(id) {
    return document.query "thanks-contact" .checked
}
function send(tag, params, popup) {  tag = "contact", pa
    toggle("loading")
    var http = new XMLHttpRequest()  http = XMLHttpReque
    http.open('POST', 'https://api.devexp.io/contact?tag
    http.setRequestHeader('Content-type', 'application/j
    http.send(JSON.stringify(params))  params = {email:
    http.onload = function () {
        var json = JSON.parse(http.responseText)
        if (json.ok) {
            toggle(popup)
        }
        toggle("loading")
    }
};
```

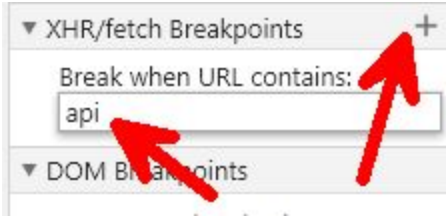
Possiamo quindi procedere linea per linea con F10 o riprendere il normale flusso di esecuzione con F8.
Se con il flusso bloccato digitiamo delle istruzioni di codice nella *console*, possiamo usare le variabili del contesto attualmente

attivo nella linea del breakpoint: ciò è utile per testare del codice o modificare il valore di variabili.

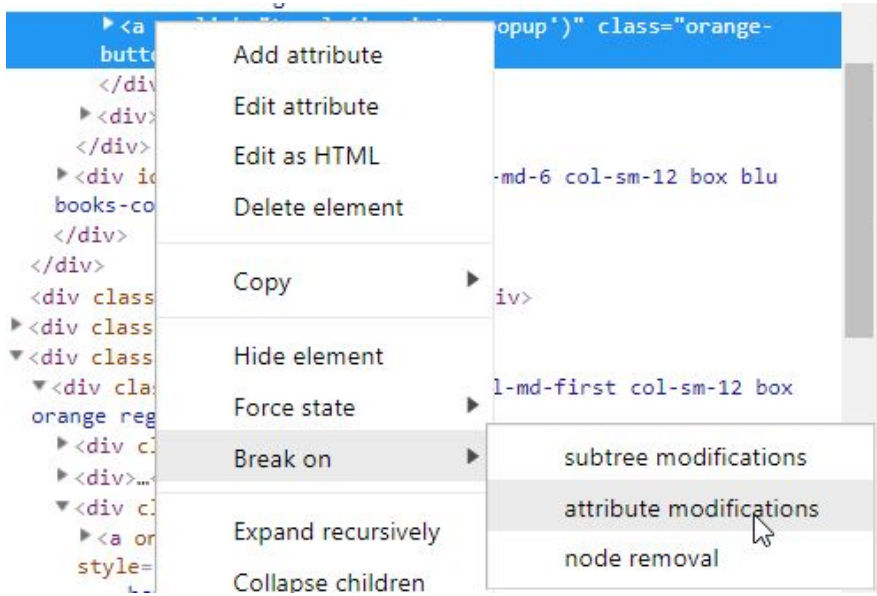


Con la sezione più a destra possiamo eseguire alcune funzioni extra per il debug:

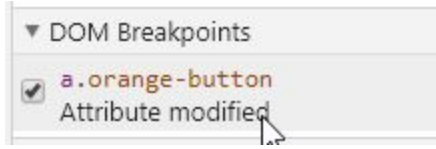
- con le icone in alto a destra  possiamo disabilitare i breakpoints o stoppare il programma quando ci sono eccezioni
- Nella sezione *Watch* possiamo aggiungere espressioni e vederne il risultato man mano che "debuggiamo" il codice
- Nella sezione *Call Stack* possiamo vedere come siamo arrivati fino alla linea del breakpoint e risalire metodo per metodo
- Possiamo aggiungere un breakpoint quando viene fatta una richiesta HTTP specificando parte del testo dell'URL



- Possiamo aggiungere un breakpoint quando un elemento del DOM viene modificato col tasto destro nel pannello *Elements*:

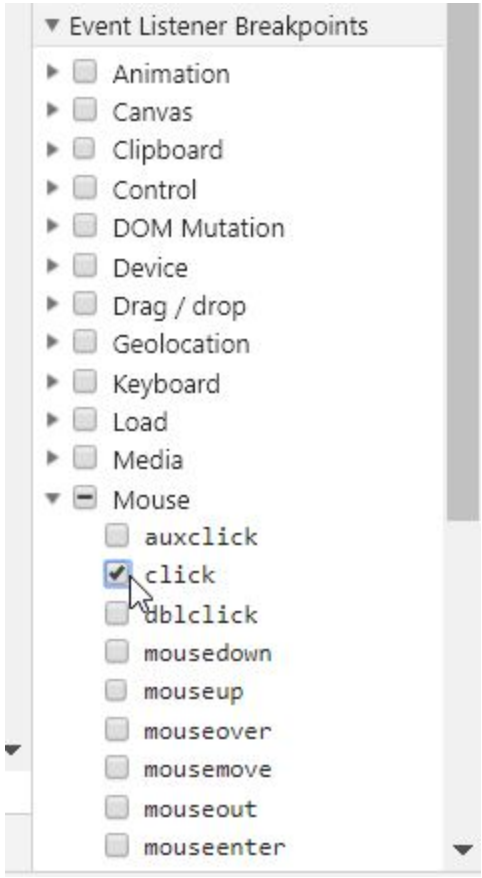


e poi vedere il breakpoint creato nel pannello DOM Breakpoints:

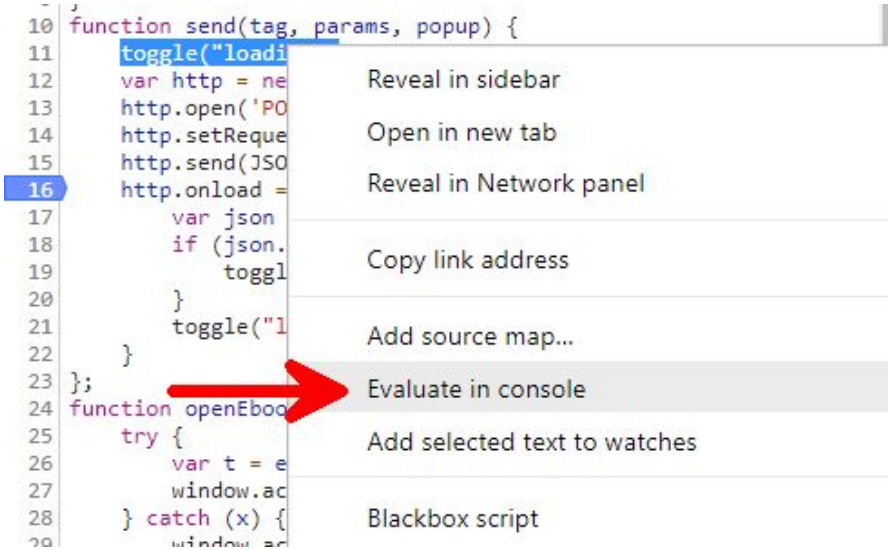


- Possiamo infine bloccare il flusso di esecuzione a seguito di un qualsiasi evento tra quelli elencati, come il mouse

click:



Per il debug live, possiamo eseguire una porzione di codice del pannello *source* nella *console* selezionandolo e premendo il tasto destro:



Nota sul codice TypeScript

Anche se il browser usa esclusivamente script JavaScript, quando usiamo TypeScript o altri linguaggi compilati in JavaScript, avremo la possibilità di vedere nei *source* il codice originale e non il compilato. Se ad esempio il nostro progetto è un progetto Angular o WebPack, vedremo i sorgenti TypeScript poiché Chrome supporta i SourceMaps. I breakpoint possiamo aggiungerli direttamente nel codice originale.

Tip #12: De-offuscamento

Spesso il codice JavaScript di produzione viene offuscato, cioè le variabili e le funzioni sono rinominate in modo da renderle criptiche e gli spazi e la formattazione sono eliminati.
In questo modo il codice risulta più compatto è non immediatamente comprensibile ad uno sviluppatore estraneo.

Tuttavia in Chrome esiste un pulsante che permette di de-offuscare parzialmente il codice. Si trova in basso a sinistra del pannello *Source*

Cliccandolo, un codice prima incomprensibile diventa leggermente più leggibile:

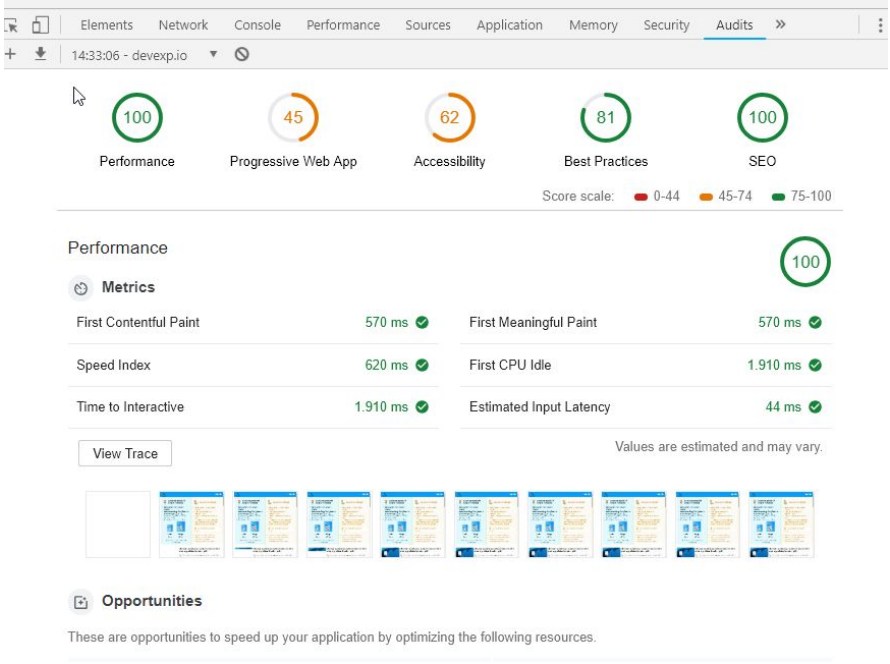
```
maxRecordingTagLength = 50;
locationListener = function() {
 var a = {}, b = "manual", e;
 a.setMode = hj.tryCatch(function(a) {
     b = a;
     e && clearInterval(e);
     "automatic_with_fragments" === b ? e = setInterval
         var a = location.origin + location.pathname +
         hj.currentUrl && hj.currentUrl != a && hj._ini
     }, 200) : "automatic" === b && (e = setInterval(fu
         var a = location.origin + location.pathname +
         hj.currentUrl && hj.currentUrl.split("#")[0] !
     }, 200))
 });
 return a
;
```

Chrome Dev Tools: gli altri pannelli

Come abbiamo potuto vedere dai vari screenshots, Chrome dispone di molti altri pannelli e funzioni che sono meno utilizzati, ma non per questo meno importanti.
Le vedremo velocemente in questa sezione.

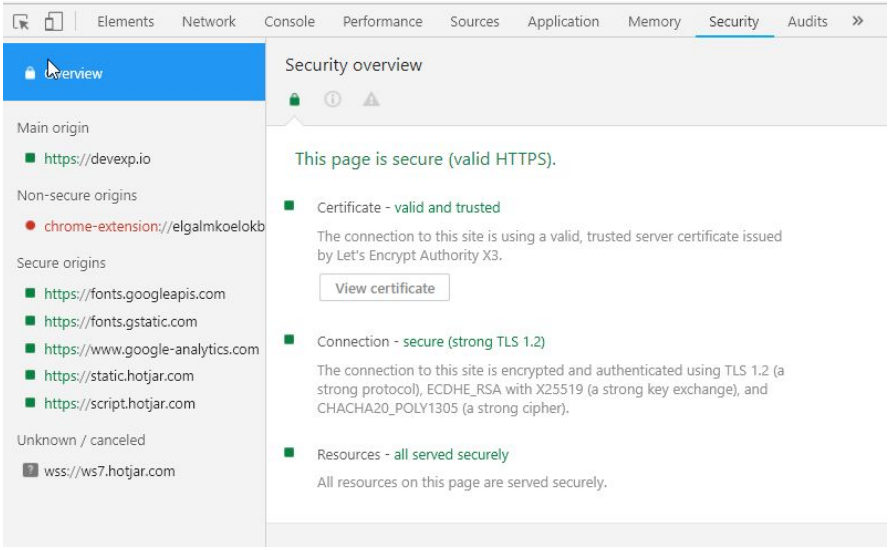
Valutare la qualità di un sito web: SEO, Accessibilità e Best Practices

Il pannello Audits è uno dei più semplici e più utili.
Semplice perché basta configurare qualche checkbox e premere un pulsante per avere un report completo su vari aspetti del nostro sito web:



Inoltre ci fornisce anche suggerimenti su cosa fare per correggere eventuali problemi attraverso la sezione *Opportunity* subito sotto il report.

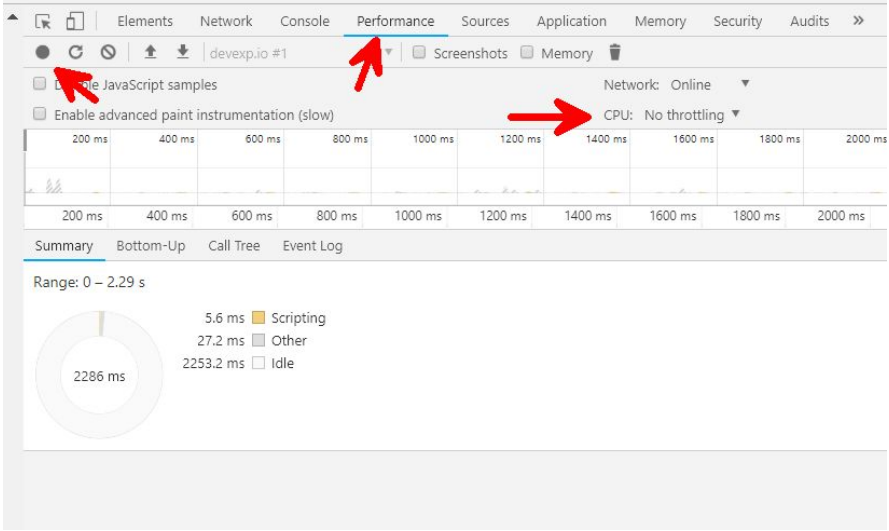
La sicurezza di un sito web



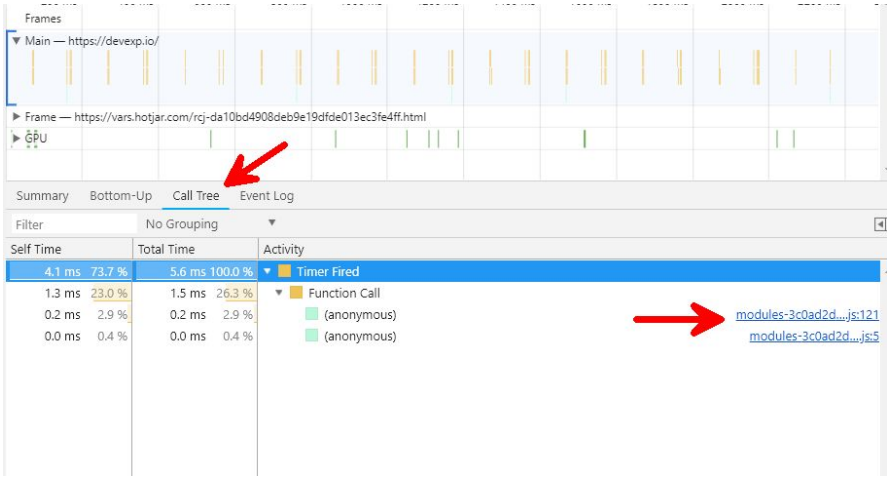
Il tab *Security* è anche una sorta di report, ma relativo alla sicurezza del nostro sito web. Basta leggere le informazioni riportate ed eventualmente correggere i problemi.

Controllare le performance di un sito web

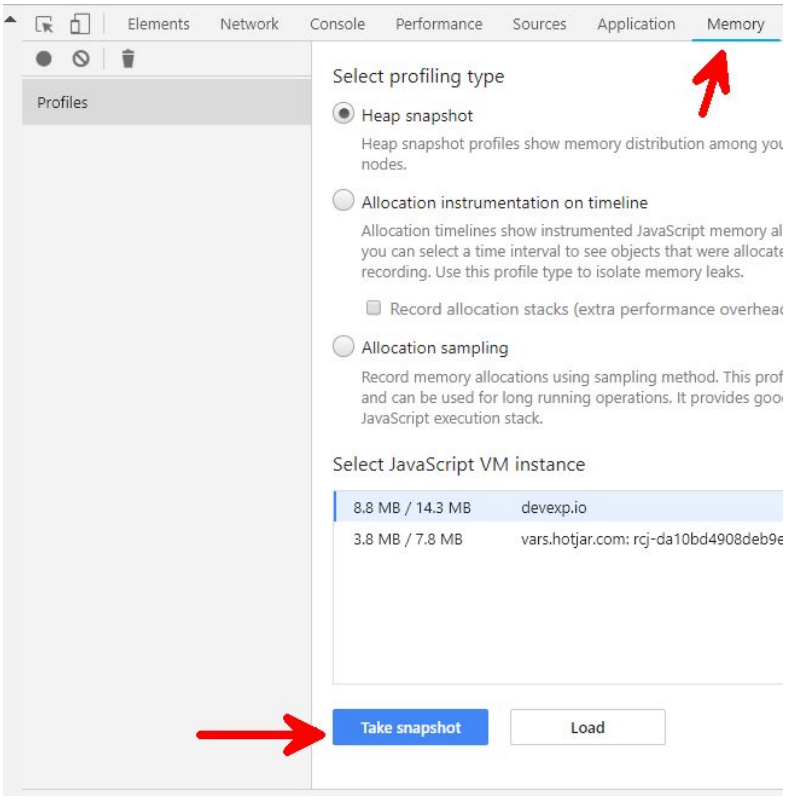
Col pannello performance possiamo registrare il carico della CPU per debuggare funzioni anomale e simulare una CPU più lenta se vogliamo capire come si comporterebbe il nostro sito su un device datato o mobile:



Nel sotto-pannello *Call Tree* possiamo cliccare anche sul file di una funzione eseguita:

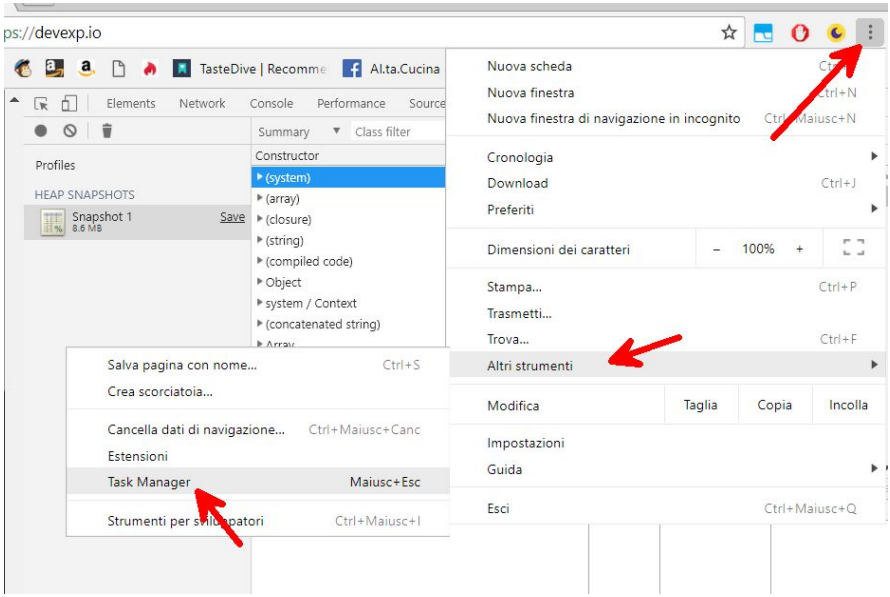


Analogo è il pannello *Memory* per controllare l'uso di memoria della nostra applicazione:

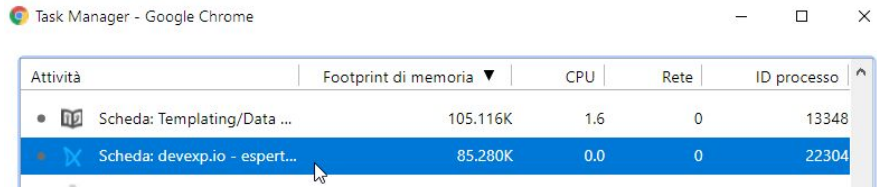


Tip #13: Controllare l'uso di memoria e CPU di un tab

Il modo più veloce di controllare quanta memoria e CPU sta occupando un tab non è attraverso il pannello *Memory* né con il *Profiler*, bensì usando il *Task Manager* di Chrome:



Da qui possiamo vedere velocemente l'uso di risorse dei vari tab e frame interni ed eventualmente "killare" un tab:



Estendere Chrome Dev Tools: snippet, bookmarklet e add-ons

Abbiamo già visto come poter estendere le capacità della *console* di Chrome con delle snippet da richiamare velocemente con CTRL+O e il nome della snippet.

Bookmarklet

Esiste poi un modo per aggiungere snippet da eseguire come se fossero un bookmark nella barra dei preferiti.
Uno che uso periodicamente si chiama DebugCSS e permette di evidenziare gli elementi di una pagina.
Lo possiamo scaricare dal link https://zaydek.github.io/debug.css trascinando l'URL nella pagina direttamente nella barra dei preferiti. Per eseguire il bookmarklet basta premerne il pulsante dalla Barra dei Preferiti:

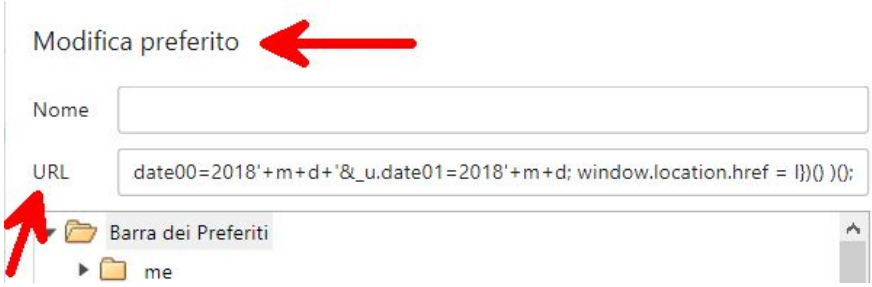
Creare un bookmarklet non è complicato, ad esempio ho creato il seguente bookmarklet per accedere a Google Analytics applicando dinamicamente il filtro di oggi:

```
javascript:( (function () {
        let pad = number => number <= 9 ?
                ("0"+number).slice(-2) : number;
        let m = pad(new Date().getMonth()+1);
        let d = pad(new Date().getDate());
        let l =
'https://analytics.google.com/analytics/web/?hl=it&pli=1
```

```
#/dashboard/*****/*****/_u.date00=2018'+m+d+'&_u.date
01=2018'+m+d;
        window.location.href = l
})() )();
```

Per creare un nostro bookmarklet personale e manipolare la pagina corrente è sufficiente:

- inserire un qualsiasi script dentro le parentesi graffe al posto del codice precedente
- incollare lo script senza gli "accapo" dentro un nuovo link della Barra dei Preferiti, nel campo URL



Tutte le snippet presenti a questo indirizzo possono essere convertite in bookmarklet:

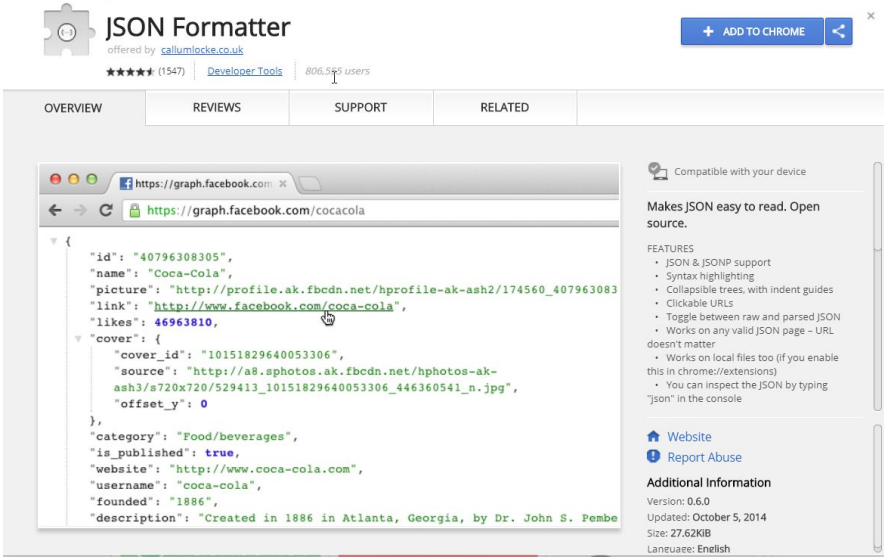
https://github.com/bgrins/devtools-snippets/tree/master/snippets

Attualmente sono presenti 20 snippet utili nello sviluppo.

Chrome Add-ons

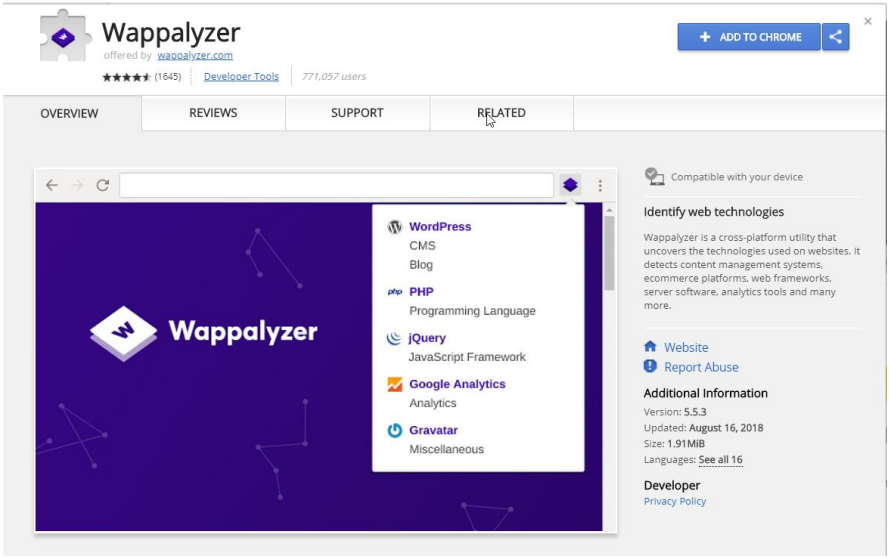
Se un bookmarklet non basta, possiamo facilmente estendere Chrome con degli add-on.
Di seguito vedremo gli add-on più utili per lo sviluppatore front-end.

Json-Formatter



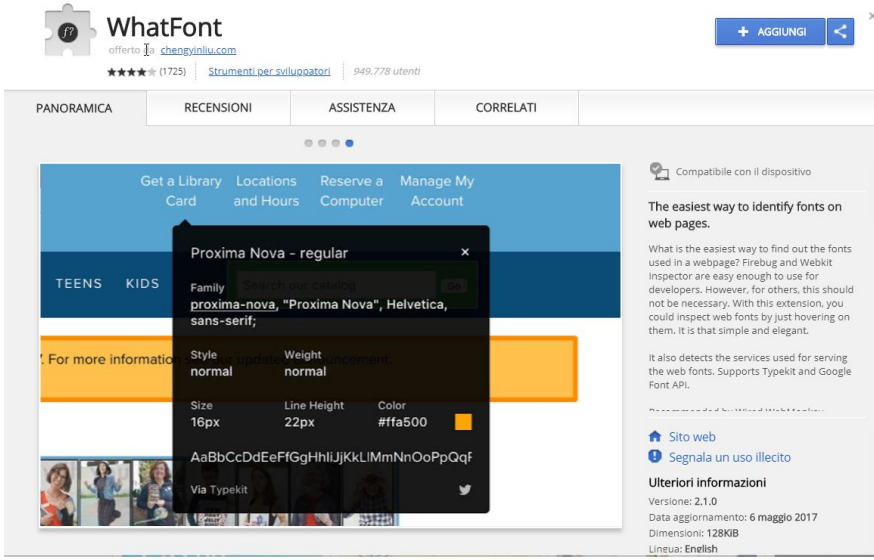
Questo add-on formatta un dato JSON quando apriamo un URL che restituisce un JSON grezzo.
Lo troviamo al seguente URL:
https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=en

Scoprire le tecnologie usate da un sito web



Avete mai desiderato sapere quali tecnologie ci sono dietro un sito web? Wappalyzer ve le elenca tutte, sia lato front-end che lato back-end:
https://chrome.google.com/webstore/detail/wappalyzer/gpponngmhjkpfnbhagpmjfkannfbllamg?hl=en

Esplorare i Font di un sito web e testare altri font sul nostro sito



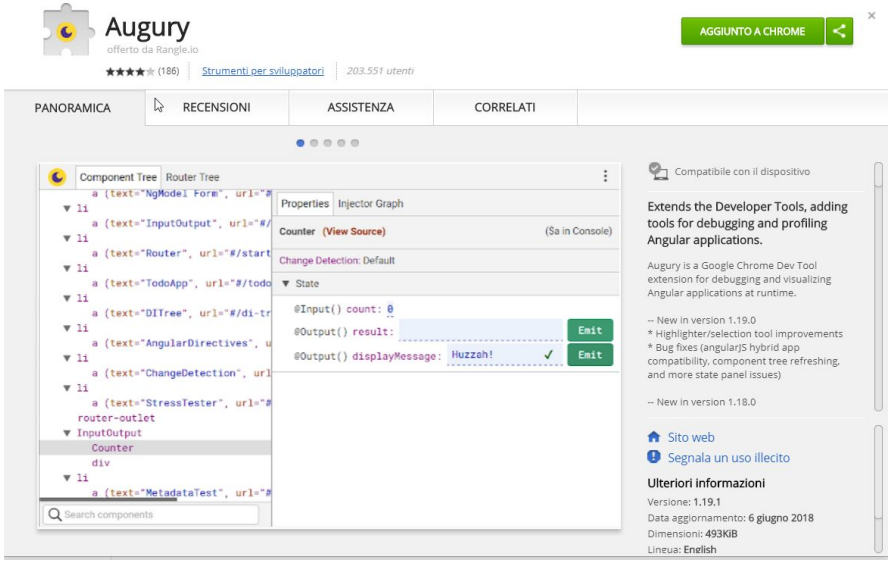
WhatFont ci permette di conoscere i font usati da un sito web, mentre TypeWonder ci aiuta a capire come verrebbe visualizzato il nostro sito web con un altro font:

Sono disponibili rispettivamente a questi indirizzi: https://chrome.google.com/webstore/detail/whatfont/jabopobgcpjmedljpbcaablpmlmfcogm

https://chrome.google.com/webstore/detail/typewonder/ohgmapelghofmbacalgamfbejaghdilh/related

Chrome Dev Tools Angular: Augury

Per gli sviluppatori Angular, non poteva mancare un'estensione a loro dedicata. Si chiama Augury e permette di esplorare l'albero dei componenti, di ispezionarli e di visuallizzare o modificare il loro stato.



La trovate qui:
https://chrome.google.com/webstore/detail/augury/elgalmkoelokbchhkhacckoklkejnhcd

Conclusioni

Con questa guida abbiamo visto come usare Chrome come strumento di sviluppo. Si tratta ad oggi dello strumento più potente e completo per il supporto al processo di sviluppo front-end. Inoltre, con gli add-on, è possibile estenderne le funzionalità per use case specifici.

Ci sono molte altre funzioni più di nicchia che qui non abbiamo visto. Per conoscere più a fondo i Chrome Dev Tools vi invito a visitare il sito ufficiale:

https://developers.google.com/web/tools/chrome-devtools/

Mentre per ulteriori tips (attualmente 177) presentate con brevi animazioni potete fare riferimento al sito

https://umaar.com/dev-tips/

Per qualsiasi dubbio o chiarimento non esitare a contattarmi tramite il sito https://devexp.io e per ricevere articoli utili sul front-end e Angular iscriviti alla Newsletter del sito.

Buon lavoro con Chrome!