Come scrivere codice JavaScript performante ed elegante

Al giorno d'oggi tante persone conoscono JavaScript, un linguaggio molto particolare da un punto di vista sintattico e semantico. Questo è molto utilizzato per la programmazione client-side per rendere interattiva una pagina HTML, anche se da un paio di anni si è affermato perfino nella programmazione server side grazie a node.js e, adesso, anche grazie a io.js, sua potenziale "controparte".

Malgrado la sua potenzialità, il linguaggio JavaScript è spesso sottovalutato dai programmatori, che preferiscono copiare/incollare il codice o affidarsi completamente a framework come jQuery che non sono quasi mai la soluzione adatta al problema, rischiando solo di appesantire e rallentare la nostra applicazione.
Bene, in questo articolo vi darò delle linee guida per scrivere codice JavaScript leggibile e bello da vedere.

Stile

Come nella scrittura esistono regole stilistiche e non si inizia mai una frase con un "però", **anche nella programmazione esistono convenzioni simili che variano da linguaggio a linguaggio**. Queste possono essere definite dal leader del team di sviluppo o essere definite da guru del linguaggio, me in ogni caso è sempre una ottima idea seguirle. Partiamo dalle basi con **l'indentazione**.
Per chi non sapesse cosa sia, l'indentazione è la combinazione di spazi e tab che rende leggibile il codice sorgente "livellandolo" e dandogli un senso di profondità indipendentemente dal flusso che tale porzione di codice segue.
La coding convention più utilizzata per l'indentazione è la tabulazione di 2 spazi, invece di una tabulazione. Questo cosa significa? Significa che al posto di premere tab quando si entra dentro un blocco, si dovranno utilizzare due spazi o, se il vostro editor lo permette, premere tab facendo in modo che inserisca due spazi automaticamente.
Tuttavia anche in questo caso molte aziende e team di sviluppo utilizzano 4 spazi, come per esempio fa Mozilla.
Adesso vedremo elencati tutti i più comuni casi in cui uno stile conciso e standard aiuta nella lettura del codice.

Dichiarazione delle variabili

Per dichiarare una variabile bisogna utilizzare SEMPRE **la keyword var** e, se la variabile è una stringa, utilizzare apici singoli. Se ci sono più variabili da dichiarare nello stesso momento, conviene omettere la ripetizione della keyword var e concluderle con una virgola mandandole a capo; l'ultima, invece, si concluderà con un punto e virgola.

Uso corretto:

Uso non corretto:

Per le costanti invece è consigliato scrivere tutte le lettere maiuscole:

Mentre nella dichiarazione di oggetti si consiglia di utilizzare sempre una scrittura ordinata ed elegante, per esempio:

E non:

Ricordandosi anche che le chiavi non devono essere definite con le doppie virgolette: non stiamo scrivendo JSON!
Inoltre, vorrei aprire una piccola parentesi sulla virgola alla fine dell'ultima chiave nella dichiarazione di oggetti: non va inserita, quindi nel primo esempio è corretto e nel secondo no!

Blocchi e condizioni

Le **condizioni nei blocchi** si impostano lasciando uno spazio dall'istruzione alla parentesi e inserendo la condizione all'interno delle parentesi senza lasciare spazi tra la parentesi di apertura e quella di chiusura.
Una volta chiusa la parentesi si lascia uno spazio e si inserisce la parentesi graffa nel caso servise (se nel blocco viene eseguita solo una linea di codice si può omettere).
Importante: è vitale non mandare mai a capo la parentesi graffa! Deve sempre stare sulla stessa linea della condizione.

Esempio corretto:

Esempio non corretto:

Naming delle funzioni

Esistono diverse notazioni che i programmatori adottano per dare nomi convenzionali alle variabili, le più famose sono la **Camel Case** e la **Underscore Case**.
La Camel Case si basa sull'unire le diverse parole che comporrebbero una variabile e mettere a

ognuna la prima lettera maiuscola, eccetto ovviamente la prima:

Al contrario, il seguente non è un corretto uso di Camel Case:

Per la cronaca quel tipo di notazione con la maiuscola all'inizio è nota come Pascal Case.
Nella Underscore Case invece si scrivono tutte le parole in minuscolo e si separano con un underscore. Per esempio la variabile precedentemente indicata diventerebbe:

In JavaScript si usa prevalentemente la Camel Case, dato che la maggior parte delle funzioni built-in del linguaggio adotta tale notazione, per esempio conoscerete funzioni come:

Funzioni e callback anonime

JavaScript offre tantissimi metodi per definire le funzioni, le quali possono distinguersi tra di loro in ambiti sincroni o asincroni.
Esistono tuttavia delle pratiche comuni per definire le funzioni e renderle più leggibili ed eleganti possibile.
Prima di tutto, per il naming delle funzioni valgono gli stessi aspetti delle variabili, utilizzando quindi il Camel Case e la keyword var, ed evitando quindi cose come:

ma scrivere

Per richiamare automaticamente una funzione, è meglio evitare di dichiararla e poi chiamarla, ma bensì adottate la seguente sintassi:

Un altro consiglio è quello di scrivere sempre delle funzioni sintetiche (se necessario dividere in più funzioni il contenuto di una sola) e che abbiano il prima possibile valori di ritorno. Non fate caso alla faccenda della programmazione strutturata, preferiamo una programmazione elegante e performante.
Per le funzioni si deve lasciare uno spazio tra la keyword *function* e le parentesi, i parametri invece non devono avere spazi tra la prima lettera e l'ultima, e bisogna anche lasciare uno spazio tra la virgola e l'ennesimo parametro.

Tips & Tricks

Inoltre esistono dei semplici trucchetti per risparmiare linee o caratteri inutili e dare un tocco di eleganza al codice, vediamone alcuni:

diventa:

Poi:

diventa

e infine:

diventa

Conclusione

Per concludere vorrei ricordare che scrivere codice che funzioni è importante, ma è altrettanto importante scrivere codice leggibile ed elegante. Una dimostrazione matematica, per quanto arduo sia stato ottenerla, è considerata maggiormente se si risolve in modo elegante: la stessa cosa vale per il codice. Programmare è un'arte, non sottovalutiamola!