SoundJS: l'audio sul web non è mai stato così facile

> «Voglio riprodurre file audio nella mia pagina web con un controllo dettagliato del suo flusso, pur utilizzando una risorsa semplice e veloce, cross-browser e molto documentata.»

Qualunque web developer, una volta che si trova alle prese con siti o applicazioni interattive, videogame e/o animazioni che necessitino di integrare anche effetti auditivi, oltre alle solite risorse puramente grafiche, si sarà posto la precedente prerogativa.
Quasi sicuramente, le prime soluzioni che vengono in mente sono:

- Utilizzare le Web Audio API
- Utilizzare Flash
- Utilizzare il tag di HTML5

In tutte queste opzioni c'è un fattore che non rispetta la nostra richiesta iniziale.
Le Web Audio API, per quanto potenti esse siano, sono un po' ostiche per i developer più inesperti e la loro messa a punto può risultare faticosa per chi tenta un approccio senza aver mai avuto confidenza con esse.
L'utilizzo di Flash, invece, è sconsigliato dato che non è supportato da piattaforme come iPhone, iPad, iPod Touch, quindi non soddisferà la richiesta di essere cross-browser.
Il tag audio di HTML5, per quanto possa essere comodo e ben supportato è molto limitato nella gamma di funzionalità a sua disposizione.

Come fare allora? La soluzione al nostro problema è **SoundJS**: una libreria JavaScript sviluppata da gskinner e utilizzata per gestire al meglio il controllo di contenuti audio nelle nostre applicazioni.
SoundJS è basato su un sistema di plugin che utilizza tutte le risorse elencate in precedenza per colmare reciprocamente con una le lacune delle altre.
I plugin disponibili sono il **WebAudioPlugin**, **l'HTMLAudioPlugin** e il **FlashAudioPlugin**.
Adesso vedremo, attraverso un'applicazione di esempio, quanto è semplice utilizzare SoundJS per dar vita alla nostra applicazione!

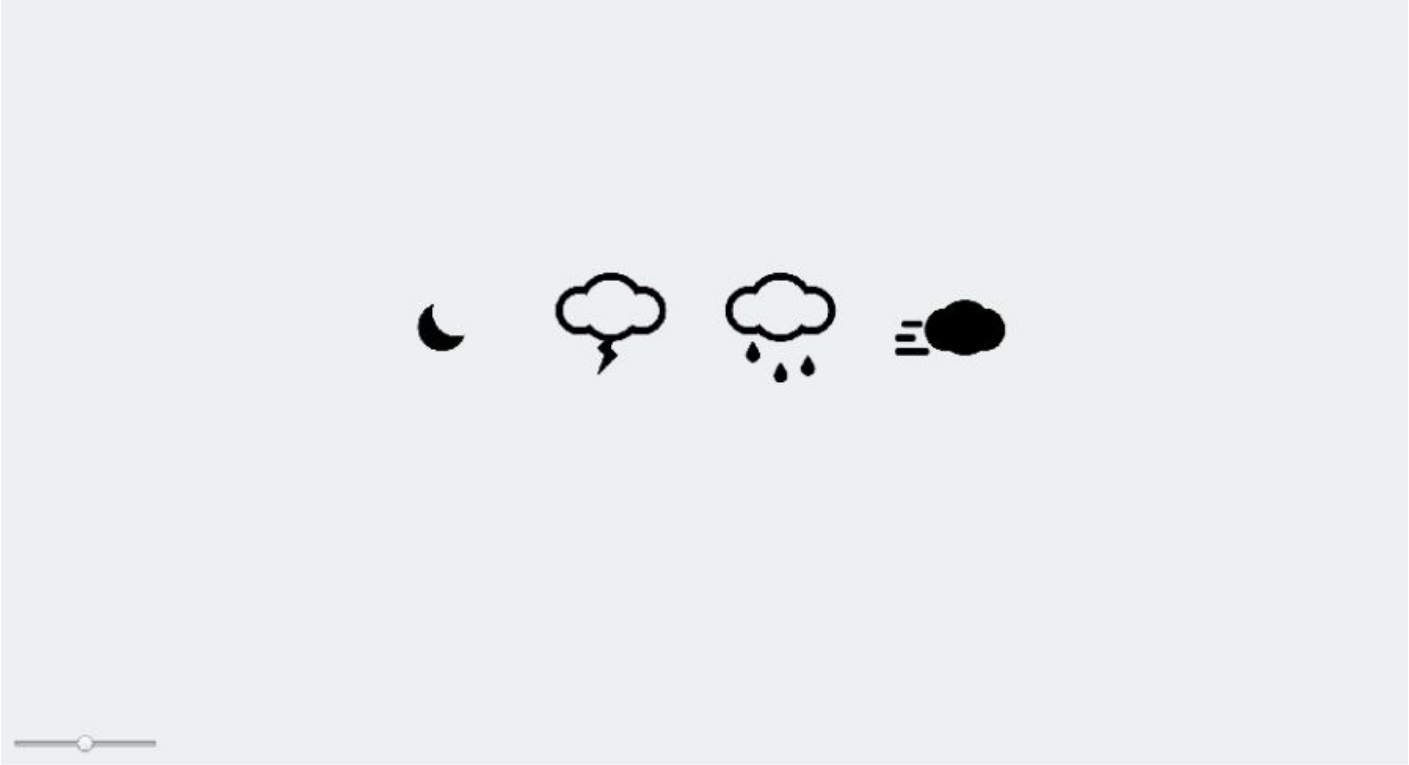
Briefing

Come applicazione di esempio realizzeremo un clone primitivo di **Noisli**, il famoso sito web in cui è possibile ascoltare e fondere i suoni della natura (vento, pioggia, fuoco) per creare un ambiente rilassante.
Per prima cosa dobbiamo preparare l'ambiente in cui la nostra pagina opererà, quindi scarichiamo il file contenente la libreria di SoundJS da GitHub e spostiamolo in una cartella chiamata **scripts**.
Per rappresentare i diversi ambienti di cui riprodurremmo il suono utilizzeremo Meteocons, un set di icone realizzato da Alessio Atzeni disponibili in questa pagina.
Una volta scaricato l'archivio, ne servirà solo il contenuto della cartella **Font-Face**, che dovrete

spostare nella nostra cartella chiamata **fonts**.
Adesso serviranno, ovviamente, i contenuti multimediali da riprodurre che vi fornisco subito io a questo indirizzo. Questi andranno spostati in una cartella che chiameremo **sounds**.
Alla fine non ci resta che creare la nostra *index.html*, fino ad avere una struttura di file come la seguente:

fonts/
sounds/
scripts/
style/
index.html

L'aspetto finale dell'applicazione sarà il seguente:



Adesso che abbiamo tutto l'ambiente pronto per realizzare la nostra applicazione di prova possiamo iniziare a mettere mano al codice!

PARTE 1: HTML

Partiamo, come è logico, dalla struttura della parte HTML dell'applicazione, che sarà la seguente:

Bene, come possiamo vedere includiamo il file CSS che si occuperà dello stile della nostra pagina, la libreria di SoundJS e il cuore dell'applicazione: **main.js**.
Poi possiamo notare le 4 icone che rappresenteranno i suoni della natura: l'attributo data-icon è utilizzato per definire quale icona di Meteocons verrà visualizzata nel tag.
Infine, ci sarà la barra del volume, con cui l'utente potrà alzare o abbassare a suo piacimento il volume dei suoni.

PARTE 2: CSS

Per rendere gradevole l'aspetto della nostra applicazione, un po' di CSS è sempre gradito!
Vediamo in seguito il breve codice che si occupa di tale compito.

Anche questo, come potete notare, è molto breve, ma ha la sua importanza!
Attraverso la regola font-face importiamo la webfont Meteocons, poi definiamo lo stile delle icone e il loro colore una volta che verranno selezionate e verrà assegnata la classe active.
L'ultima regola è un trick usato da Meteocons per utilizzare l'attributo data-icon.
Adesso la nostra applicazione è pronta per diventare effettivamente viva, siete pronti?

PARTE 3: JAVASCRIPT

Finora abbiamo solo parlato in termini di design, è a questo punto che vediamo la vera parte logica e le potenzialità di SoundJS.
Mettiamoci all'opera creando subito la funzione changeVolume che, come potete vedere dal codice HTML, verrà eseguita all'evento di change dell'input range del volume. Il codice è il seguente:

Molto semplice, vero? Non fa altro che prendere il parametro passato (che va da 0.0 a 1, quindi 0 è muto e 1 è il massimo volume) e passarlo come parametro al metodo setVolume di createJS.
Levata questa spina dal fianco iniziamo a vedere un pezzo di codice alla volta!
Prima di tutto carichiamo i file .mp3 dalla cartella sounds e rendiamoli utilizzabili da SoundJS; ecco il codice:

Per prima cosa definiamo, mediante l'attributo alternateExtensions, l'estensione da cercare nel caso quella passata non sia trovata. Nel nostro caso, se nomefile.mp3 non verrà trovato, verrà cercato nomefile.wav; lo segnalo per scrupolo, anche se spero che, avendo seguito le istruzioni, non ce ne sia il bisogno!
L'istruzione successiva serve per eseguire la funzione fileLoaded (che andremo a definire in seguito) ogni qualvolta che un file è stato caricato.
Infine carichiamo i nostri file e il gioco è fatto!

Abbiamo notato che all'evento di fileload viene eseguita la funzione fileLoaded, perché non esaminarla allora? Eccola qui:

Anche questa, come si può vedere è molto semplice: selezioniamo l'icona di cui ci è stato passato l'ID (night, thunder, rain o wind) e bindiamo su di essa all'evento click l'esecuzione della funzione playSound (anche questa verrà vista in seguito), passando come parametro l'ID dell'icona che ci servirà per eseguire il file mp3 richiesto.
Infine ecco il sorgente della funzione playSound:

In questa funzione non facciamo altro che controllare se è presente la classe active nell'icona e quindi se è già stata cliccata.
Se è presente la togliamo e stoppiamo la riproduzione di quel suono, se invece non lo è, la appendiamo e riproduciamo il file. Che ne pensate? È veramente semplice come vi ho promesso? Bene, le funzioni sono finite, vediamo il sorgente JavaScript completo:

Adesso abbiamo finalmente utilizzato SoundJS per creare un clone di Noisli!

Conclusioni

L'articolo è terminato, spero di essere stato chiaro e di non avervi annoiato! Adesso sapete che SoundJS è una risorsa valida e spero che vi divertiate a utilizzarlo il più possibile nei vostri progetti. Trovate il sorgente a questo indirizzo! Alla prossima!