JavaScript – La mia prima animazione (parte 2)

Nel primo articolo di questa guida abbiamo visto come creare un'animazione in JavaScript molto semplice e per di più, se ricordi bene, avevamo notato che quest'ultima presentava due potenziali problemi: il ritardo e lo spostamento erano costanti.

Ora il nostro compito sarà quello di evitare questi due problemi, creando uno script che sia cross-browser e che venga visualizzato (quasi) nello stesso modo da tutti.

Puoi vedere il risultato finale di ciò che andremo a fare cliccando qui.

Per riuscire nel nostro intento ecco punto per punto ciò che dobbiamo fare:

1. Definire la durata totale dell'animazione in ms (millisecondi)
   il suo contenuto verrà salvato dentro la variabile time
2. Ricavare l'ora locale dalla macchina del visitatore
   il suo contenuto viene salvato all'interno della variabile start
3. Ad ogni cambio di fotogramma verranno effettuati i seguenti passaggi
    1. Calcolare nuovamente l'ora locale
       l'ora verrà memorizzata nella variabile now
    2. Calcolare il tempo trascorso dall'inizio dell'animazione fino al momento attuale e dividere il risultato per la durata totale
       (now - start) / time
    3. Passare il valore ottenuto dal passo precedente ad una equazione che restituisce la distanza
       il valore restituito dall'equazione verrà applicato come avanzamento nel prossimo fotogramma

Questo è quanto; non lasciarti scoraggiare dalle parole, vedrai che creare la funzione sarà più facile a farsi che a dirsi.

Quanto appena detto potrebbe essere tradotto in codice così:

Come puoi vedere dal codice precedente, l'oggetto **animate** accetta 3 argomenti:

1. **easing**
   questo parametro rappresenta il tipo di effetto che verrà applicato alla nostra animazione; l'effetto sarà una function che tramite un'operazione matematica ci restituisce il valore dello spostamento da applicare in base all'effetto desiderato (ad. es. lineare, elastico etc.)
2. **time**
   rappresenta la durata totale dell'animazione ed è espressa in millisecondi
3. **callback**
   funzione che verrà eseguita ad ogni fotogramma

All'inizio della nostra funzione vengono definite le seguenti variabili

1. **ant**
   contiene il valore percentuale dell'avanzamento dell'animazione rispetto alla durata finale
2. **done**
   rappresenta lo stato dell'animazione, è di tipo Boolean e ritorna true quando l'animazione è finita
3. **_this**
   contiene un richiamo allo scope della funzione principale, in modo da renderlo disponibile anche all'interno delle sue funzioni (init e next)
4. **start**
   contiene l'ora di inizio dell'animazione, e viene ricavata dalla macchina dell'utente

Inoltre, come puoi vedere, la funzione animate contiene due metodi:

1. **init**
   è il metodo che viene chiamato ad ogni fotogramma.
   Ogni volta che init viene chiamato si controlla (tramite il **metodo next**) se il tempo trascorso dall'inizio dell'animazione è o non è minore della durata totale (definita in **time**):
   se è minore, il metodo next ritorna il valore percentuale da applicare all'animazione, se invece non lo è, ritorna false e quindi l'animazione termina.
2. **next**
   controlla se il tempo trascorso dall'inizio dell'animazione è minore oppure no rispetto alla durata totale definita nella variabile time.
   Ritorna false se il tempo trascorso è maggiore, in caso contrario ritorna il valore

percentuale da applicare all'animazione (questo valore lo ricava dalla funzione (**easing**) alla quale passa come argomento la percentuale di tempo trascorso rispetto alla durata totale)

Il metodo **init**, come abbiamo già detto controlla che il valore ritornato da **next** sia diverso da *false*; quando questo accade, **init** chiama la funzione **callback** e le passa come argomento il valore restituito da **next**.
I valori restituiti dal metodo next vanno da 0 a 1 e rappresentano il valore percentuale da applicare all'animazione.

Dopo aver chiamato la funzione callback, init richiama nuovamente se stessa.

Durante una delle chiamate ricorsive di **init**, il valore restituito da **next** sarà false. Quando ciò accadrà, **init** chiamerà per ultima volta il callback passando il valore massimo che si aspetta: 1

Ecco come si presenta il codice finale

Come puoi vedere, oltre all'oggetto **animate** ora abbiamo due nuove funzioni: **sposta** e **lineare**.

La funzione **sposta** è quella che si occupa di "dare il via" all'animazione e come argomento accetta l'elemento al quale verrà applicata l'animazione.

All'interno della funzione sposta vengono definiti il punto d'inizio (**inizio**) ed il punto d'arrivo (**fine**); queste due variabili sono fondamentali al funzionamento dell'animazione poichè è grazie a loro che riusciamo a calcolare il totale percorso (**fine - inizio**), e quindi calcolare la nuova posizione dell'elemento (**o**) alla quale applicheremo la percentuale (**p**) che ci viene fornita dal metodo **init** quando chiama il callback.

Dopo aver definito il punto d'inizio e d'arrivo viene istanziato l'oggetto animate, passando i tre argomenti richiesti:

1. **tipologia di easing** (lineare)
2. **tempo di durata in ms** (1000)
3. **callback**
   in questo caso, il callback è una funzione che applica all'elemento da animare il suo nuovo valore ottenuto dall'equazione spiegata in precedenza

Una volta istanziato l'oggetto viene richiamato il metodo init e subito dopo, viene "distrutta" l'istanza (settando t a null), poichè da li in poi, non ha più motivo di esistere.

La funzione lineare, rappresenta invece l'effetto di easing che verrà applicato all'animazione, in questo caso, l'effetto è lineare, infatti la funzione ritorna il valore percentuale (**p**) e quindi

l'animazione si evolve a velocità costante; utilizzando funzioni che invece elaborino questo valore si potrebbero creare dei diversi effetti, ad esempio di decelerazione, elastico e così via.

All'interno del metodo onload è stato inserito un piccolissimo timer (1000 ms) per far iniziare l'animazione un secondo dopo che il DOM sia stato caricato (questo passaggio è facoltativo).

Per capire meglio questi processi matematici, vi lascio il link con le equazioni di Robert Penner, tramite il quale potete capire meglio il loro funzionamento.

Ecco di seguito, una lista di effetti semplici da utilizzare con l'animazione :

Quad_easeIn e Quint_easeIn da jQuery easing plugin.

demo Quad_easeIn
demo Quint_easeIn

Elastico leggero

demo Elastico leggero

Decelerazione

demo Decelerazione

Conclusioni

In questa guida è stato spiegato il funzionamento delle animazioni in JavaScript, abbiamo visto quali potrebbero essere i problemi principali di un'animazione basilare e abbiamo quindi visto come risolverli.

Da questa base si potrebbe perfino creare un piccolissimo framework per crearsi le proprie animazioni, magari soltanto per divertimento.

Gli articoli di questa guida

1. JavaScript, la mia prima animazione: capire il funzionamento e conoscere i principali limiti.
2. JavaScript, la mia prima animazione: creare delle animazioni basilari ed evitare i problemi ad esse legati.