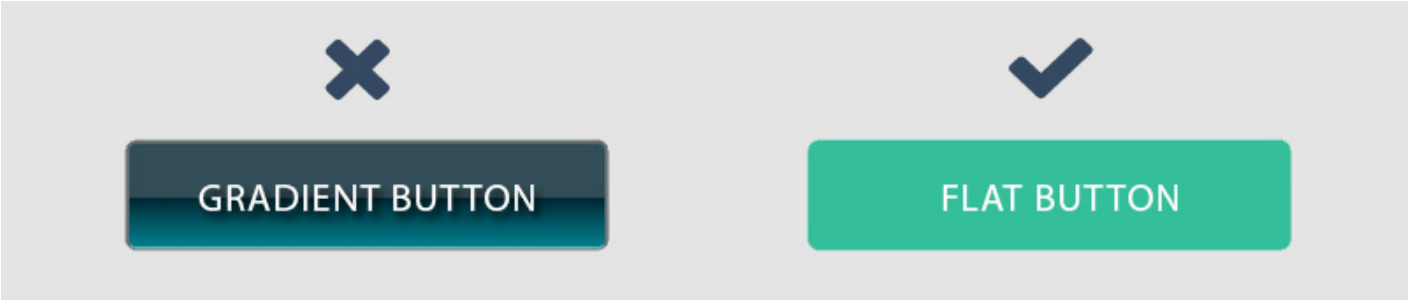
HTML Mobile App: 5 Consigli per aumentare le performance

Negli articoli precedenti (HTML5 Mobile App: transizioni ed animazioni CSS3 parte 1 e parte 2) abbiamo visto come poter realizzare **applicazioni HTML5** e **Javascript** con interfacce che sfruttano animazioni **CSS3**, e come poter controllare quest'ultime grazie agli eventi che abbiamo a disposizione.

Oggi ti voglio mostrare alcune tecniche da affiancare a quelle già analizzate per migliorare l'esperienza utente e renderla più vicina, in termini di prestazioni e reattività, a quelle sviluppate nativamente.

1. Limita l'utilizzo di ombre e riempimenti gradienti e sfrutta l'accelerazione hardware

Se possibile evita di usare **proprietà CSS** "**pesanti**" come **colori gradienti** e **ombre**, le quali rallentano notevolmente il processo di **rendering** che la memoria centrale del dispositivo deve affrontare (fortunatamente il flat design ci è d'aiuto :)).



Smartphone e **tablet** presentano generalmente caratteristiche **hardware** molto meno potenti rispetto a quelle desktop, e risulta piuttosto facile provocare un **memory "overload"** causato dalla presenza di un numero troppo elevato di processi da gestire simultaneamente (il classico scenario in cui si verifica il **crash dell'applicazione).**

Una pratica molto diffusa ed efficace è quella di **delegare parte dei processi di rendering alla GPU** (la scheda grafica), favorendo la composizione dei **livelli grafici,** ovvero la trasformazione di **gruppi** di elementi **DOM** in immagini **bitmap.**

Si tratta di quella tecnica denominata **hardware acceleration** (o accelerazione hardware), che viene attivata tramite determinate proprietà CSS3.

Per approfondire il discorso dell'accelerazione hardware, sapere su quali proprietà agire ai fini delle prestazioni e capire come poter sfruttare la creazione dei livelli grafici, ti consiglio di leggere

questo articolo

- Web e Mobile App Fluide con i livelli grafici e l’hardware acceleration

2. Semplifica la struttura DOM del documento

Una delle caratteristiche fondamentali di qualsiasi single page app (e quindi di ogni applicazione mobile sviluppata in HTML5) è la generazione del markup HTML lato-client, quindi tramite Javascript, utilizzando all’occorrenza dei sistemi di javascript templating.

Si tratta di un aspetto molto importante per contribuire alla velocità dell’interfaccia, dal momento che invece di generare codice HTML direttamente dal server, quest’ultimo è utilizzato soltanto per lo scambio dei dati.

Tuttavia bisogna fare attenzione a non abusare nella generazione del markup: un numero elevato di elementi DOM potrebbe scatenare troppi processi di rendering allo stesso tempo, portando ad un sovraccarico della memoria, ovvero lo stesso tipo di situazione che abbiamo visto nel punto uno.

Ecco alcune tecniche che possono contribuire ad alleggerire e rendere più semplice la struttura HTML della tua applicazione:

- Usa il minor numero possibile di elementi DOM
- Minimizza le modifiche e gli interventi sugli elementi DOM presenti nel documento: ognuna di esse potrebbe far ripetere il rendering per l’elemento interessato
- Rimuovi dal documento gli elementi non utilizzati al momento, e genera nuovi elementi quando hai bisogno di mostrarli all’utente
- Se necessario, modifica un elemento DOM prima di aggiungerlo al documento: ciò permette di richiamare il processo di rendering soltanto quando l’elemento viene inserito

Ecco un esempio che mette a confronto due porzioni di codice Javascript, che fanno essenzialmente la stessa cosa, ma con una ripercussione diversa sulle prestazioni:

3. Non utilizzare troppi framework esterni

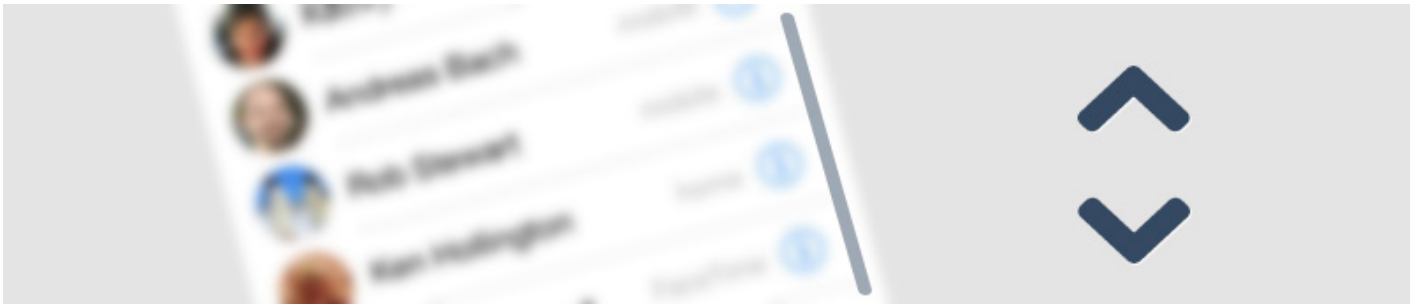
Molte volte capita di includere nel progetto diversi tipi di **framework** solo per usufruire di una parte delle caratteristiche di ognuno (effetti CSS3, elementi UI, foglio di stile ecc), rallentando notevolmente i tempi di caricamento iniziale e quelli durante la creazione di nuovo markup.

È un errore comune aggiugere riferimenti che rimangono completamente inutilizzati o quasi,

Come parte di fogli di stile (o fogli di stile interi), file javascript o qualunque altro tipo di risorsa.

Prima di pubblicare il tuo progetto **controlla sempre ciò che rimane inutilizzato**, evita di includere interi framework e, se ne utilizzi soltanto una parte o se proprio hai bisogno di un determinato componente di quel framework, assicurati di personalizzarne il download: molte librerie permettono infatti di poter scegliere cosa includere sul pacchetto da scaricare, come nel caso di Twitter Bootstrap, Modernizr, jQuery UI, Animate.css e molti altri.

4. Usa lo scrolling nativo



Quando possibile cerca di applicare sempre lo scrolling nativo per mostrare i lunghi contenuti, incorporando il framework per lo scrolling javascript soltanto se il primo non è supportato.

Così facendo avrai due vantaggi principali:

1. **Eliminare il caricamento iniziale** della libreria Javascript usata per lo scrolling quando non necessaria
2. Usufruire delle **prestazioni native** del dispositivo, che sicuramente sono molto più performanti e fluide di quelle offerte da qualunque framework javascript

Ecco l'esempio di codice per abilitare lo scrolling nativo iOS su di un elemento con id ***#scrolling-wrapper***, posizionato in modo assoluto nel documento:

5. Evita di andare troppo in profondità nella struttura del DOM ed usa la proprietà display:none invece che visibility:hidden

Evita di inserire troppi elementi DOM nidificati in modo da creare una struttura gerarchica troppo complessa: modificando infatti anche solo una proprietà CSS3 per l'elemento più interno è possibile riscatenare il processo di rendering per tutti gli elementi che lo contengono, risalendo fino al primo livello.

Credimi, ciò può avere un impatto negativo incredibile nelle performance della tua HTML5 Mobile App, rendendola lenta o provocandone addirittura il crash.

Nel punto 2 ti ho consigliato di eliminare gli elementi DOM che non vuoi visualizzare in un dato momento, per poi ricrearli quando ne hai bisogno; tuttavia ciò non è sempre possibile.

Molte volte capita che per qualche motivo non puoi rimuovere completamente un componente dell'interfaccia, ma allo stesso tempo abbiamo visto che lasciare nel documento troppi elementi può causare un sovraccarico della memoria centrale.

In questi casi resta soltanto una soluzione: **nascondere l'elemento con la proprietà *display:none***, a discapito della proprietà visibility:hidden.

Il browser tratta gli elementi resi invisibili con la seconda proprietà come se fossero ancora visibili (rimangono cliccabili e mantengono un ingombro nel documento), pertanto "resta in ascolto" per eventuali modifche agli stili o alla posizione, ripetendo quindi il processo di rendering.

Un elemento nascosto invece dalla proprietà display viene completamente ignorato dal browser,

che quindi non si preoccupa di fare attenzione ad eventuali modifiche CSS.

Conclusioni

Come avrai notato, focalizzarsi sulla riduzione del numero di volte in cui il processo di rendering viene ripetuto per ogni elemento DOM costituisce la base per la realizzazione di una Applicazione HTML Mobile (e non) fluida e reattiva.

Proprio per questo, se non lo stai già facendo, ti consiglio vivamente di **testare continuamente le performance di ciò che stai sviluppando**, utilizzando gli ottimi strumenti per lo sviluppatore messi a disposizione dai vari tipi di browser, che permettono di capire quale parte può essere migliorata e resa più “leggera”.

Se hai qualche consiglio o tecnica da aggiungere condividila lasciando un tuo commento!