- Joe Casabona -

# Responsive Design

## con WordPress

Rendere responsive la navigazione, le immagini, i widget e le funzionalità blog >>

Tutorial per creare slider di foto, mappe, gallerie di prodotti >>

Utilizzare framework per temi responsive >>

Loop, plugin e shortcode >>

*pro DigitalLifeStyle

*Ai miei genitori, Louis e Marie,*
*per il loro continuo supporto.*
*E a Joe e Jean Rizzi;*
*i loro consigli, la loro gentilezza e la loro pazienza*
*mi hanno aiutato ad arrivare dove sono oggi.*

*pro
Digital*LifeStyle

# Responsive Design con WordPress

**Joe Casabona**

LSWR

Responsive Design con WordPress

**Autore**: Joe Casabona
**Traduzione**: Francesco Caccavella

**Collana**:

*pro Digital Life Style

**Editor**: Marco Aleotti
**Progetto grafico**: Roberta Venturieri

**ISBN**: 978-88-6895-095-8


Via Spadolini, 7 - 20141 Milano (MI) - www.lswr.it

Finito di stampare nel mese di luglio 2014 presso "Rotolito Lombarda" SpA, Pioltello (MI)

# Sommario

# Prefazione

Il 20% di tutti i siti Web sono oggi gestiti attraverso WordPress, ed è prevedibile che WordPress gestirà un sito su quattro, tra quelli lanciati nel 2014. Ho creato il mio primo sito Web basato su WordPress nel 2005, prima che lo facessero tutti i ragazzi *cool*. Sono molto orgoglioso di essere uno sviluppatore veterano e un esperto di un prodotto usato da milioni di persone in tutto il mondo.

Da quando ho lanciato quel primo sito nel 2005, ho scritto un libro sullo sviluppo per WordPress e ne ho pronti altri per l'uscita nella primavera del 2014. Ho anche contribuito ad altri libri, ho scritto articoli per pubblicazioni online come *Smashing Magazine* e *net Magazine* e insegno sia nelle Università sia online. Sono stato anche speaker in conferenze in tutto il mondo e in una di queste ho incontrato Joe Casabona.

Mi sono sentito onorato quando Joe mi ha chiesto di scrivere la prefazione di questo libro perché sapevo che sarebbe stato un ottimo lavoro. Joe ha un vero talento nel trasformare complicate soluzioni in semplici istruzioni passo per passo. WordPress è stato progettato per essere semplice: semplice da installare, semplice da configurare e semplice da estendere. Tuttavia, può risultare ancora un po' difficile da comprendere per i designer alle prime armi e per gli sviluppatori che cercano di estenderne le funzionalità di base.

Queste difficoltà mi hanno spinto nel 2012 a scrivere il mio libro *Web Designer's Guide to WordPress: Plan, Theme, Build, Launch* ed è esattamente il motivo per cui Joe ha scritto il suo libro quest'anno. Siamo entrambi sviluppatori veterani che vogliono far crescere la comunità di WordPress. Il modo migliore per farlo è quello di aiutare a educare la comunità e condividere le nostre esperienze e le nostre conoscenze su un prodotto che usiamo ogni giorno. Questo è quello che ha fatto Joe con *Responsive Design with WordPress*. È un libro ben strutturato con un gran numero di ottimi esempi.

So, come professore in due Università del Rhode Island, che questo librò potrà integrare in modo perfetto il curriculum dei miei corsi. Le lezioni, gli esempi e anche le domande alla fine di ogni capitolo vi aiuteranno a sviluppare dei solidi fondamenti di

WordPress e del Web Design Responsive. Durante la lettura, inoltre, svilupperete un tema per WordPress in modo da rafforzare le vostre competenze a mano a mano che proseguirete nel libro.

Nondimeno, apprenderete due competenze contemporaneamente. Conoscerete WordPress e, allo stesso tempo, maturerete un'esperienza specifica sul Web Design Responsive. Questo approccio non solo aiuterà a rafforzare le vostre abilità in entrambe le aree, ma vi renderà anche degli esperti in una nicchia molto redditizia.

Come ho detto prima, WordPress gestirà il 25% di tutti i siti web lanciati nel 2014. Questo significa che un nuovo sito su quattro avrà bisogno di uno sviluppatore che conosca WordPress. Di più: a partire da quest'anno le informazioni saranno ricevute più sui dispositivi mobile che sui computer tradizionali. Se non disponete nel 2013 di forti competenze nel Web Design Responsive, sicuramente ne avrete bisogno nel 2014 e oltre.

A mio parere, non c'è modo migliore per imparare una competenza che farlo da soli. Questo libro è il modo migliore per imparare, contemporaneamente, WordPress e il Web Design Responsive. Ottimo lavoro, Joe!

*Jesse Friedman*

# Introduzione

Ho avuto il mio primo dispositivo portatile quando ero una matricola al liceo. Era un Palm m100 e lo adoravo. Non faceva granché, e, beh, a 13 o 14 anni non lo potevo usare per fare molte cose. Ma avere un computer in tasca? Pazzesco! Il risultato fu che lo portavo con me dovunque andassi (e può darsi che mi sia stato sequestrato una o due volte dopo averlo usato in classe).
Poi sono passato a un Compaq iPAQ, che era basato su Windows e aveva uno schermo a colori. Ho pensato che fosse ancora più pazzesco. Su questo dispositivo potevo eseguire *veri* programmi. Quando stavo facendo visita ai college nei primi anni 2000 chiedevo anche se vi fosse il Wi-Fi, quando il Wi-Fi stava appena diffondendosi. Pensavo a tutte le cose incredibili che potevo fare con un piccolo computer che aveva il suo pennino (stilo) e che poteva stare in tasca. Comunque, dopo un po' mi sono ritrovato a volere qualcosa di più. Cosa che mi ha portato al mio primo smartphone: il Palm Treo 650 (**Figura 1**).

**Figura 1 -** Il Treo 650. A volte mi manca ancora.

Con questo telefono facevo di tutto: chiamare, scattare foto, sincronizzare il mio account Google. E aveva anche un browser primitivo chiamato Blazer. Potevo addirittura navigare sui siti web dal mio cellulare!
Da allora, naturalmente, il panorama della telefonia portatile è cambiato. L'iPhone ha introdotto un browser completo per dispositivi mobile, compatibile con tutto, dai CSS a JavaScript. L'iPhone non ha però risolto un problema: il problema del piccolo schermo. È qui che il Web Design Responsive entra in gioco.
Forse ne avete già sentito parlare. È un principio apparentemente molto noto in questo periodo. Tante persone diverse – sviluppatori, designer, agenzie e utenti – si stanno facendo domande sulla sua natura. E perché non dovrebbero? È un bene essere aggiornati su ciò che soddisfa un mercato in rapida crescita. Il Web Design Responsive è diventato una di quelle cose che le persone analizzano quando visitano un sito (ridimensionare una pagina web ha sostituito il "controlla nel codice sorgente il layout a tabelle").
Se state progettando un sito web, non avete in fin dei conti alcun controllo su come il sito viene visualizzato; non potete decidere dove è visualizzato, con che cosa è visualizzato o la connessione con la quale è visualizzato. Questa incertezza potrebbe spaventare qualcuno, ma per me (e scommetto anche per voi) è tutto il contrario. Adoro risolvere questo tipo di problema. Ciò non vuol dire che non esistano preoccupazioni. La necessità è creare un sito web che sia facile da usare su dispositivi mobile, ma che faccia anche dire "wow" quando viene visualizzato sul desktop. Questo, in fin dei conti, è ciò che è il Web Design Responsive.
Anche WordPress è grande. Su questo sistema si basano milioni di pagine web. Centinaia di milioni. Come avete letto nella prefazione, un sito web su quattro di quelli lanciati nel 2014 sarà basato su WordPress. WordPress fa molte cose al nostro posto e, allo stesso tempo, ci permette di fare altro e altro ancora. Dunque: che c'entra WordPress con il Web Design Responsive? Beh, a quanto pare, può essere davvero utile per la creazione di temi responsive e ha un gran numero di funzionalità pronte all'uso che noi, come sviluppatori, possiamo sfruttare per creare migliori siti responsive. Ed è proprio quello che vi mostrerò.

## A chi è rivolto questo libro?

Mi piacerebbe dirvi che questo libro è stato scritto per chiunque voglia sviluppare siti WordPress, ma per toccare il vero motivo per cui ho scritto questo libro, ho bisogno di fare alcune ipotesi su di voi, cari lettori.
In primo luogo, si suppone che abbiate una conoscenza di HTML, CSS, PHP, JavaScript e MySQL. Ho anche presupposto che abbiate una certa familiarità con WordPress – lo

avete installato, lo avete utilizzato, forse avete anche creato un tema. Infine, presumo che abbiate usato in qualche modo un server: dovreste almeno conoscere la struttura delle directory di WordPress e saper usare FTP o SFTP.
Questo libro è, dunque, per sviluppatori web e sviluppatori WordPress che vogliono approfittare di ciò che WordPress mette a disposizione, per creare eccezionali siti web responsive. In questo libro analizzeremo una vasta gamma di argomenti e di tecniche per convertire gli elementi dei siti web in elementi responsive di un tema WordPress.
Non mancherò, tuttavia, di fornire alcune informazioni di base. Nel primo capitolo, daremo uno sguardo più da vicino al Web Design Responsive: che cos'è, quali sono le sue origini e quali le migliori pratiche. Poi, ci sarà una breve panoramica sullo sviluppo di un tema per WordPress; in questa parte analizzeremo alcuni dei principali componenti di un tema WordPress: i file importanti, il Loop, i tipi di post personalizzati, i plugin e altro ancora. Poi, ci spingeremo nella parte davvero divertente.
Il cuore del libro – creare un tema responsive per WordPress – è diviso in tre parti. Nel Capitolo 3 vedremo le principali tecniche responsive e come integrarle nel tema WordPress. I Capitoli 4 e 5 esamineranno i componenti specifici di un sito WordPress, tra cui la navigazione, le immagini, i commenti, i widget, gli archivi e i plugin.
Chiuderemo il libro dando un'occhiata, nel Capitolo 6, ai framework per temi responsive e ai temi figlio, e facendola seguire, nel Capitolo 7, da una sezione stile "libro di ricette", ricca di tutorial per lo sviluppo responsive.

## Perché ho scritto questo libro?

Quando mi è venuta l'idea di questo libro, c'erano un sacco di cose che fluttuavano nella mia testa. Il Web Design è sempre in evoluzione; WordPress è sempre in evoluzione. Le migliori pratiche di un paio di anni fa sono cambiate in entrambi i campi: è importante rendere disponibili queste informazioni.
Nella comunità degli sviluppatori web c'è grande attenzione al principio di "fare responsive responsabilmente" ("doing responsive responsibly", una frase coniata da Scott Jehl); questa frase delinea l'idea che quando si parla di responsive non si parla solo delle dimensioni dello schermo. Nella comunità di WordPress sta inoltre prendendo piede la tendenza a rimuovere le funzionalità dai temi (caratteristiche come gli slider o i tipi di post personalizzati che si basano sul contenuto). Dal momento che molti sviluppatori web stanno probabilmente lavorando, allo stesso tempo, con il responsive design e con WordPress, ho voluto creare un luogo in cui queste cose vengano discusse insieme.

## Convenzioni di scrittura del codice

Prima di tutto, qualsiasi codice che si incontra nel libro verrà presentato in due modi. Potrebbe essere del codice in riga, come questo: `<?php echo "Hello World!"; ?>` o potrebbe essere del codice in un suo proprio blocchetto, tipo questo:

```
function hello_world(){
     $s= "Hello World";
     return $s;
}
print hello_world();
```

In entrambi i casi, dovreste essere in grado di riconoscerlo rapidamente. Per quanto riguarda gli standard di scrittura del codice, il Codex di WordPress ne indica un bel po' (http://rwdwp.com/16). Farò del mio meglio per aderire a questi standard.
Per indicare che il codice stampato in pagina è estratto da una porzione di codice più ampia (nel quale, cioè, è stata saltata qualche istruzione tra la prima riga elencata e la successiva), utilizzo i puntini di sospensione (. . .).
Un paio di cose che vorrei sottolineare: utilizzerò markup HTML5, ma non faremo nulla con le caratteristiche più avanzate di HTML5, come i Web Sockets o le API di archiviazione.
Nella maggior parte dei casi, i CSS useranno le `.classi` invece degli `#id`. Questo dovrebbe produrre codice CSS più pulito, eliminando la necessità di selettori molto specifici. Tutto il codice CSS sarà formattato come il seguente:

```
.class-name{
     color: #FFFFFF;
     background: #000000;
    }
```

Si noti l'uso di trattini (-) al posto delle notazioni a cammello o degli underscore e il fatto che la parentesi di chiusura è rientrata. Questo rende più facile la lettura del codice CSS, soprattutto quando ce n'è molto.
Al contrario, i nomi delle mie funzioni PHP useranno sempre l'underscore (_) e saranno preceduti dal prefisso mf_, così: `mf_get_featured_images ()`.

**NOTA** Cercate del testo come questo per note e suggerimenti.

Infine, a volte i limiti di layout di una pubblicazione a stampa ci obbligano a prolungare singole righe di codice su più di un rigo. In questi casi, si vedrà una freccia (→) a indicare che queste righe di codice non devono essere spezzate quando le si utilizza. Se hai la versione digitale di questo libro, potresti notare che a volte il codice si spezza imprevedibilmente nel testo. In tal caso, è importante affidarsi, per accuratezza, ai file scaricabili (www.rwdwp.com).

## Altre note

C'è molto codice nel libro. La maggior parte delle volte indicherò dove è possibile reperirlo. Se non lo faccio, tutto il codice utilizzato è disponibile sul sito del libro, www.rwdwp.com, così come su GitHub. Troverete anche un elenco di tutti gli URL brevi, con i siti a cui rindirizzano.

A mano a mano che scriverete il codice trovato nel libro, noterete che non faccio menzione dei test prima del Capitolo 6: è importante fare dei test su almeno un paio di dispositivi, soprattutto se si pensa di utilizzare queste tecniche in siti pronti per essere messi online (e io spero lo facciate).

Infine, ho la tendenza a usare un sacco di acronimi, che vengono di solito sciolti nel contesto della frase. Nel caso in cui non lo siano, qui ci sono quelli più comuni:

- RWD (*Responsive Web Design*): Web Design Responsive.
- WP: WordPress.
- RESS (*Responsive Design + Server Side Components*): Responsive Design + componenti lato server.
- Il Codex: il Codex di WordPress (o la documentazione delle API).

# Ringraziamenti

Vorrei ringraziare le seguenti persone, senza le quali questo libro non sarebbe nelle vostre mani:

- Stephen Mekosh, non solo perché è un ottimo redattore di tecnologia e un buon amico, ma anche perché è stato la prima persona ad avermi mostrato i CSS e WordPress;
- Michael Nolan per avermi dato la possibilità di scrivere questo libro e per avermi accolto nella Peachpit Press/New Riders;
- Nancy Peterson ed Eric Schumacher-Rasmussen per i consigli e l'assistenza, per l'editing, per avermi mantenuto lungo la giusta rotta e per i nostri colloqui settimanali;
- Joanne Gosnell e Scout Festa per, rispettivamente, il copyediting e la correzione delle bozze e perché hanno fatto sembrare buona la mia padronanza della grammatica e della lingua inglese;
- il team di design della Peachpit Press per aver reso meraviglioso questo libro;
- Jesse Friedman per la splendida prefazione e per le gentili parole;
- Jason Coleman, Stephanie Leary, Lisa Sabin-Wilson e Pippin Williamson per avermi lasciato utilizzare le loro idee su WordPress e sull'arte di scrivere un libro;
- la mia famiglia e gli amici, in modo particolare i miei genitori, i miei fratelli Phil, Mike e Rob, Dave Redding, Rob McManus, Matt Wren e la mia meravigliosa fidanzata, Erin Holman.

# 1

# Che cos'è il Web Design Responsive?

A essere onesti, probabilmente non siete qui per imparare cosa sia il Web Design Responsive (RWD). Sapete già che cosa è e sapete anche perché è nato. Il web è in continua evoluzione; mentre scrivo, ci sono più di 6000 diverse risoluzioni dello schermo sui soli dispositivi Android e i nostri siti hanno bisogno di adattarsi! Innumerevoli libri sono stati scritti sull'argomento da quando questa idea è emersa nel 2009.
Potreste aver letto uno di quei libri o un post di un blog sul tema, scritto da uno dei tanti autori, o, forse, avete già implementato un vostro personale layout responsive. Voi siete qui per imparare come sfruttare WordPress per migliorare i vostri layout responsive.
Per la stessa ragione, non possiamo iniziare da metà strada. Avrete bisogno di imparare qualche concetto di base prima di entrare a fondo nell'argomento. Fidatevi di me, lo sto facendo per me e per voi.
Questo capitolo si inoltra prima nella storia del Web Design Responsive, partendo dal post di un blog che ha lanciato il tutto, per poi spostarsi sulle migliori pratiche. Vedrete le migliori metodologie per gestire la determinazione e la creazione dei breakpoint, diverse considerazioni sul RWD e lo stato attuale dei dispositivi (allerta spoiler: c'è un'enorme diversità tra i dispositivi). Per questo motivo, dovreste avere almeno familiarità con HTML e CSS e con la teoria che è alla base delle griglie fluide.

## Le origini del Web Design Responsive

Ethan Marcotte ha coniato il termine *Responsive Web Design* nel suo articolo dallo stesso titolo per la webzine *A List Apart* (http://rwdwp.com/1). In quell'articolo afferma:

> *Piuttosto che adattare design indipendenti a ciascuno di un numero sempre crescente di dispositivi web, possiamo trattarli come facce della stessa esperienza. Possiamo progettare per un'esperienza di visualizzazione ottimale, ma incorporare nei nostri design tecnologie standard per renderli non solo più flessibili, ma più adattabili al medium che li visualizza. In breve, abbiamo bisogno di esercitarci nel Web Design Responsive.*

Il resto, come si suol dire, è storia. Il mobile è una parte integrante della società e le persone stanno facendo sempre più cose con i loro telefoni ... ma di questo parleremo diffusamente più avanti. Adesso ci occuperemo di come implementare il RWD e della parte forse più importante di un design responsive: i breakpoint.

## Breakpoint & media query

I breakpoint e le media query sono il sistema attraverso cui noi (tramite i CSS) diciamo al browser quando adattare il nostro layout in base alle dimensioni dello schermo. Le media query, in sé, non sono un concetto particolarmente nuovo: vengono utilizzate da un bel po' di tempo e sono supportate dalla maggior parte dei browser (**Figura 1.1**). Gli sviluppatori utilizzano le media query per applicare i CSS in base a specifiche condizioni (per esempio, solo sulle versioni per gli schermi o sulle versioni per la stampa, oppure in base alla larghezza del browser). I breakpoint sono larghezze del browser definite dallo sviluppatore: sono il punto in cui il layout cambierà. Per esempio, 1024px può essere considerato un breakpoint.

CSS3 Media Queries - Recommendation

Method of applying styles based on media information. Includes things like page and device dimensions

Resources: WebPlatform Docs · IE demo page with information · Media Queries tutorial · Polyfill for IE

Global user stats*:
Support: 87.09%
Partial support: 0.01%
Total: 87.1%

| | IE | Firefox | Chrome | Safari | Opera | iOS Safari | Opera Mini | Android Browser | Blackberry Browser | Opera Mobile | Chrome for Android | Firefox for Android |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 versions back | | | 4.0 | | | | | | | | | |
| 22 versions back | | | 5.0 | | | | | | | | | |
| 21 versions back | | 2.0 | 6.0 | | | | | | | | | |
| 20 versions back | | 3.0 | 7.0 | | | | | | | | | |
| 19 versions back | | 3.5 | 8.0 | | | | | | | | | |
| 18 versions back | | 3.6 | 9.0 | | | | | | | | | |
| 17 versions back | | 4.0 | 10.0 | | | | | | | | | |
| 16 versions back | | 5.0 | 11.0 | | | | | | | | | |
| 15 versions back | | 6.0 | 12.0 | | | | | | | | | |
| 14 versions back | | 7.0 | 13.0 | | | | | | | | | |
| 13 versions back | | 8.0 | 14.0 | | | | | | | | | |
| 12 versions back | | 9.0 | 15.0 | | | | | | | | | |
| 11 versions back | | 10.0 | 16.0 | | | | | | | | | |
| 10 versions back | | 11.0 | 17.0 | | 9.0 | | | | | | | |
| 9 versions back | | 12.0 | 18.0 | | 9.5-9.6 | | | | | | | |
| 8 versions back | | 13.0 | 19.0 | | 10.0-10.1 | | | | | | | |
| 7 versions back | | 14.0 | 20.0 | | 10.5 | | | | | | | |
| 6 versions back | | 15.0 | 21.0 | | 10.6 | | | 2.1 | | 10.0 | | |
| 5 versions back | 5.5 | 16.0 | 22.0 | 3.1 | 11.0 | | | 2.2 | | 11.0 | | |
| 4 versions back | 6.0 | 17.0 | 23.0 | 3.2 | 11.1 | 3.2 | | 2.3 | | 11.1 | | |
| 3 versions back | 7.0 | 18.0 | 24.0 | 4.0 | 11.5 | 4.0-4.1 | | 3.0 | | 11.5 | | |
| 2 versions back | 8.0 | 19.0 | 25.0 | 5.0 | 11.6 | 4.2-4.3 | | 4.0 | | 12.0 | | |
| Previous version | 9.0 | 20.0 | 26.0 | 5.1 | 12.0 | 5.0-5.1 | | 4.1 | 7.0 | 12.1 | | |
| Current | 10.0 | 21.0 | 27.0 | 6.0 | 12.1 | 6.0-6.1 | 5.0-7.0 | 4.2 | 10.0 | 14.0 | 25.0 | 19.0 |
| Near future | 11.0 | 22.0 | 28.0 | 7.0 | 15.0 | 7.0 | | | | | | |
| Farther future | | 23.0 | 29.0 | | | | | | | | | |

Note: Incomplete support by older webkit browsers refers to only acknowledging different media rules on page reload

**Figura 1.1 -** Il supporto dei browser alle media query, dal sito caniuse.com.

L'avvento dei dispositivi mobile ha spinto la necessità delle media query e dei breakpoint. D'altro canto, dobbiamo sempre tenere a mente che stiamo parlando di una tecnologia in evoluzione. Quando Marcotte ha introdotto per la prima volta il Web Design Responsive non c'erano molti dispositivi popolari; il risultato fu che emersero alcuni breakpoint "generici".
Da allora, il numero delle dimensioni degli schermi è cresciuto in modo esponenziale, sia in larghezza, sia in altezza, e sono stati avanzati diversi approcci riguardo a come vengono rappresentati e a come vengono determinati i breakpoint. Quelle che sto per delineare non sono solo considerate le migliori pratiche (al momento in cui scriviamo), ma sono anche le tecniche che tratterò in tutto il resto del libro.

**NOTA**

Quando il RWD apparve per la prima volta, emersero alcuni breakpoint "generici" abbastanza corrispondenti con i prodotti Apple. Questi breakpoint oscillavano intorno ai 320px, 768px e 1024px.

## Breakpoint basati su em

Nella sezione precedente, avete visto come sono emersi "breakpoint generici" basati sulle larghezze specifiche, in pixel, del browser. Già nel 2011, gli sviluppatori hanno scoperto che questi breakpoint generici non potevano essere la base per le migliori pratiche del RWD, per due motivi. Il primo ha a che fare con i siti web che si possono adattare non solo al dispositivo, ma anche alle impostazioni dell'utente. Nel suo articolo *The 'trouble' with Android* (I 'problemi' con Android, http://rwdwp.com/2), Stephanie Rieger spiega come sia impraticabile non solo progettare layout per dimensioni specifiche dello schermo, ma anche presumere di conoscere quali saranno le impostazioni del dispositivo dell'utente. Sebbene il browser imposti alcuni valori predefiniti per le dimensioni dei caratteri, i margini e altro, gli utenti hanno la possibilità di modificare tali impostazioni predefinite per soddisfare al meglio le loro esigenze. Se, per esempio, un utente ha difficoltà a leggere del testo di piccole dimensioni, lui, o lei, potrebbe aumentare la dimensione predefinita del testo. Se un sito web è progettato correttamente, la dimensione del carattere aumenta automaticamente con le impostazioni dell'utente. È questo il motivo per cui dobbiamo passare da breakpoint basati sui pixel a quelli basati sugli em.
In realtà, gli sviluppatori web utilizzano da anni gli em per definire le dimensioni degli elementi. Questo perché un testo con le dimensioni del carattere basate sugli em, e non sui pixel, è l'elemento-chiave per ottenere del testo flessibile che non confligga con le impostazioni del browser. I pixel sono valori assoluti: è un concetto molto importante da tenere a mente. Se le dimensioni dei caratteri (e di altri elementi) sono definite utilizzando i pixel, essi non cambieranno in base alle preferenze dell'utente.

NOTA

Fatto divertente: em non è in realtà un acronimo o un'abbreviazione di una parola, come px è per pixel. È un'unità di misura legata alle dimensioni di un tipo di carattere ed è basata sulla M maiuscola per il tipo di carattere utilizzato: "em" è la trascrizione fonetica di "M". Nei CSS, em non è più associato alla dimensione relativa di un tipo di carattere, ma è, invece, relativo alle dimensioni del browser e dell'elemento contenitore.

Utilizzando gli em per le dimensioni del carattere, queste dimensioni saranno impostate correttamente in base al design che abbiamo scelto, nonché in base alle preferenze dell'utente; quindi se l'utente esegue uno zoom con il browser, il nostro design non si scombussola. Questa stessa tecnica può essere applicata ai breakpoint per i quali, per definire il punto in cui il nostro layout cambia, invece di usare i pixel, useremo gli em. Questo garantisce che il layout del sito sia flessibile e responsive non solo in base alla larghezza del browser, ma anche in base alle impostazioni dell'utente. In altre parole, dona all'utente il pieno controllo di come visualizzare il contenuto.

## Convertire px in em e percentuali

Ricavare gli em dai px è piuttosto semplice: em rappresenta 16px, quindi 1em = 16px. Naturalmente ci sono anche molte risorse che ti aiuteranno a eseguire la conversione. La mia preferita è www.pxtoem.com. Il servizio eseguirà le conversioni in entrambe le direzioni, così come modificherà, per voi, il valore dei pixel di base. Naturalmente, scendendo nei dettagli (o cascando nei dettagli, se si vuole), il tutto non sarà così semplice, poiché gli elementi ereditano i loro stili dagli elementi genitore. Ciò significa che se la dimensione del font è 20px, il vostro valore base sarà 20px, non 16px. Ma non preoccupatevi! Ethan Marcotte offre una semplice soluzione matematica per scoprire il valore dei vostri em:

```
target/contenuto = risultato
```

Quindi, se la dimensione del font di base è 20px e si desidera che la dimensione di un carattere sia 24px, basterà eseguire il calcolo 24/20 = 1,2, quindi la dimensione del carattere sarà pari a 1.2em. Allo stesso modo, se si vuole che sia 16px, basterà calcolare 16/20 = 0,8 o 0.8em. Quando si lavora con i layout, è possibile fare lo stesso calcolo per ottenere le percentuali. Quindi, se decidi che il tuo layout debba essere basato su una griglia di 960px, l'area di contenuto sarà pari a 960px. Se desideri che l'area di contenuto principale sia 600px, otterrai una larghezza flessibile (in percentuale) calcolando 600/960 = 0,62. Moltiplicando per 100, si otterrà 62%.

Questo è perfettamente in linea con il cuore del RWD: noi non sappiamo quale dispositivo useranno gli utenti per visualizzare il nostro sito, quindi lo dovremmo progettare in modo tale che sia visualizzato correttamente su qualsiasi dispositivo.
Quindi è chiaro che i breakpoint flessibili e basati sugli em sono la strada da percorrere per garantire la massima flessibilità di un layout. Tuttavia, c'è ancora una cosa che dobbiamo fare per quanto riguarda le media query, in modo da renderle il più possibile agnostiche rispetto ai dispositivi.

## I breakpoint basati sui contenuti

Tornando al concetto di breakpoint "generici", gli sviluppatori progetterebbero per tre dispositivi o giù di lì; vale a dire: per il desktop, per l'iPad e per l'iPhone. Così, invece di bloccare il design su un solo dispositivo, lo si sta bloccando su tre dispositivi. È un miglioramento, ma un piccolo miglioramento date le centinaia di dispositivi ora disponibili per i consumatori. A causa del flusso costante di dispositivi emergenti e dell'insieme sempre più diversificato delle risoluzioni dello schermo, è un compito quasi impossibile determinare per quale schermo si dovrebbe progettare. Ecco perché quella che è ormai emersa come la pratica migliore è determinare i breakpoint non basandoli su uno specifico dispositivo, ma sui contenuti. Nel suo articolo, Rieger espone le ragioni a favore di questo tipo di progettazione.
Determinando i breakpoint in base al contenuto, invece di immaginare i dispositivi su cui vogliamo che il nostro layout venga visualizzato correttamente, cercheremo i punti nei quali il nostro layout si scombussola e lo correggeremo in quel punto. Nell'immagine della **Figura 1.2**, il mio logo/avatar comincia a consumare troppo spazio, quindi ho bisogno di aggiungere un breakpoint in quel punto, per ridimensionarlo adeguatamente.


**Figura 1.2 -** Il mio sito web quando la finestra del browser si espande.

**NOTA**

Per controllare i breakpoint uso un'estensione per Chrome chiamata Window Resizer, disponibile all'indirizzo http://rwdwp.com/3.

Le **Figure 1.3** e **1.4** mostrano un altro dei miei siti web in cui i pulsanti, in un determinato punto, non vanno oltre il 50% dello schermo, rendendo completamente disordinato il design. Quello è il punto dove dovrebbe andare un breakpoint.
In entrambi i casi, ho bisogno di controllare come appare il sito quando allargo la finestra del browser, per poi annotare i valori di em, nel punto in cui il layout cambia o viene visualizzato male.


**Figura 1.3 -** Un mio piccolo progetto marginale nella visualizzazione mobile.


**Figura 1.4 -** Lo stesso progetto esteso a circa il doppio della larghezza.

## Lo stato attuale dei dispositivi

Nelle ultime due sezioni avete visto come rappresentare i breakpoint nel modo più flessibile possibile. Ora è il momento di analizzare perché è fondamentale determinare i breakpoint in base al contenuto, piuttosto che in base ai dispositivi. Ci sono oltre 6000 risoluzioni dello schermo solo su Android – una statistica che ho sentito citare la prima volta da Luke Wroblewski in una conferenza del 2012. Una ricerca sommaria per "dispositivi mobile più diffusi" conduce a decine di post di blog, articoli e statistiche che delineano la top 10, 20, e persino 50, dei dispositivi mobile. Brighthand.com, un blog di notizie su smartphone, pubblica la classifica dei primi 41 dispositivi nei loro ultimi 5 giorni di traffico. Se guardiamo alla mia collezione di dispositivi, la maggior parte dei quali può connettersi a Internet, abbiamo:

**Tabella 1.1**

| Dispositivo | Dimensione dello schermo | Risoluzione |
|---|---|---|
| Samsung Galaxy SIV, smartphone | 5 pollici | 1920x1080 |
| Google Nexus 7, tablet | 7,02 pollici | 1920x1200 |
| Apple iPhone 4S, smartphone | 3,5 pollici | 640x960 |
| Apple nuovo iPad, tablet (terza generazione) | 9,7 pollici | 2048x1536 |
| Google TV box di Sony (prima generazione)* | 55 pollici | 1920x1080 |
| Microsoft Xbox 360* | 55 pollici | 1920x1080 |
| Sony PlayStation 3* | 55 pollici | 1920x1080 |
| Google Project Glass, proiettore a prisma ** | | 640x360 |
| Amazon Kindle, Ereader | 6 pollici | 600x800 |

* Per i dispositivi che si connettono al televisore ho usato le informazioni del mio televisore.
** Equivalente a uno schermo di 25 pollici visto da 2,4 metri.

I dispositivi sono molti e io sono solo un singolo utente! Se andassi a casa della mia famiglia, dovremmo aggiungere l'iPhone 5, l'HTC 8.X con Windows Phone, un Kindle Fire (seconda generazione) e molti altri dispositivi che posseggono i miei genitori e i miei fratelli. StatCounter Global Stats ha un grafico con le 14 più diffuse risoluzioni dello schermo di dispositivi mobile (**Figura 1.5**).
Sarebbe poco pratico cercare di progettare per tutti questi dispositivi o risoluzioni dello schermo, e sarebbe comunque impossibile prevedere ciò che il grafico mostrerà fra un anno da oggi. Questa è la ragione principale per cui la determinazione dei breakpoint in base al contenuto, e non ai dispositivi, è incredibilmente importante per raggiungere una buona esperienza dell'utente.

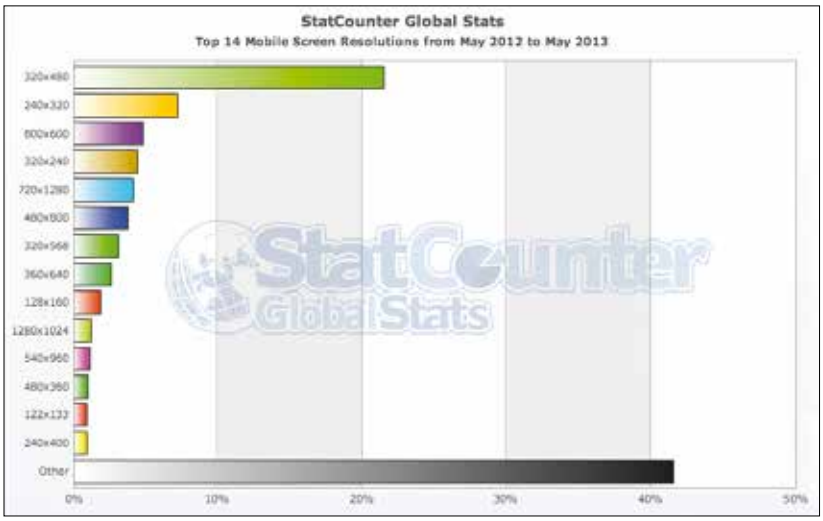


**Figura 1.5 -** Le prime 14 risoluzioni di schermo sui dispositivi mobile di StatCounter, da maggio 2012 a maggio 2013. Si noti che oltre il 40% sono nella categoria "Altro". [Fonte: http://rwdwp.com/4]

## Considerate la velocità di connessione

Questo non è un libro sul RWD in generale, né una panoramica sullo sviluppo per dispositivi mobile, ma nel concludere questa sezione voglio citare alcuni ultimi aspetti del RWD che sono importanti per creare una buona esperienza dell'utente. Dopo tutto, un eccellente design o degli eccellenti contenuti saranno invisibili all'utente se l'esperienza complessiva non è buona.

### Abbiate giudizio!

Se utilizzate librerie JavaScript, framework CSS o dell'altro codice supplementare in modo intelligente, non comprometterete l'esperienza per gli utenti. Ci sono alcune librerie che utilizzo che aggiungono valore al sito senza rallentarlo troppo. Alcune domande da porsi:

1. perché utilizzo questa libreria?
2. c'è un modo migliore per raggiungere il mio obiettivo?
3. devo usare tutto ciò che è incluso nella libreria o posso eliminare qualcosa?

Alcune finalità per cui io utilizzo librerie JavaScript e snippet:

a. supporto HTML5 per IE;
b. supporto media query per IE;
c. visualizzazione della navigazione mobile (a volte).

Ricordate: non tutti visualizzeranno il vostro sito dalla più veloce delle connessioni Wi-Fi o 4G. Gli utenti potrebbero navigare in 3G, da un Wi-Fi debole o anche da una connessione Edge. Mantenete i vostri siti web quanto più snelli possibile. Questo significa non caricare, se non è necessario, librerie JavaScript superflue o immagini enormi. Ciò è importantissimo perché, come sottolinea Brad Frost, il 74% degli utenti abbandona un sito se impiega più di 5 secondi a caricarlo. Librerie come jQuery possono risultare utili, ma non vale la pena considerarle se sono molto pesanti e non aggiungono molto valore.
Le prestazioni di un sito, tuttavia, vanno oltre le cose che abbiamo considerato. Ho visto un sacco di siti implementare CSS che nascondono, a seconda di breakpoint, determinate sezioni di una pagina, al fine di mostrare le sezioni più "ottimizzate per lo schermo". Anche se l'intenzione è lodevole, il risultato può essere negativo perché si sta costringendo l'utente a scaricare del codice o del markup supplementare.

**NOTA**

Ricorda: i CSS possono pesare parecchio. Ottimizza i tuoi fogli di stile e, quando puoi, raggruppali.

Chi utilizza uno smartphone non vedrà mai il codice HTML per il desktop, anche se lo ha appena scaricato sul suo dispositivo. Interventi come questo possono anche sommarsi a vicenda, soprattutto se si stanno nascondendo immagini o altri elementi multimediali. Questa, almeno per me, è la ragione più convincente per considerare un approccio "Mobile First" per la progettazione di siti web.

## Mobile First!

Nel suo libro *Mobile First*, Luke Wroblewski parla di come progettare siti web a partire dallo schermo più piccolo disponibile. Egli sostiene (e io sarei d'accordo) che ciò costringe a prendere in considerazione le caratteristiche e gli elementi dell'interfaccia utente più importanti, come se si stesse progettando per uno schermo piccolo. Questo influirà anche sul tipo di librerie e sugli effetti che si utilizzano.
Quando progetto un layout "Mobile First" ho meno probabilità di includere immagini extra e funzioni aggiuntive che, alla fine, consumeranno dello spazio. Quando sistemo il layout per dimensioni più ampie, questi principi di base rimangono ancora validi.

Ci sono altre ottimizzazioni da eseguire sui file per alleggerire il peso che grava sull'utente. Per quanto riguarda specificamente le immagini, e pensando ad alcune delle

tecniche che vedremo nei capitoli successivi, è possibile manipolarle attraverso strumenti che riducono le dimensioni dei file, senza perderne la qualità. Su Mac, è possibile usare ImageOptim (http://rwdwp.com/5). Su PC, c'è RIOT (http://rwdwp.com/6). (**Figura 1.6**).

**Figura 1.6 -** Un'immagine ottimizzata con ImageOptim. L'originale è a sinistra e la versione ottimizzata è a destra (formato ridotto del 10%). Foto di Philip J. Casabona.

Infine, se ci rivolgiamo ad altri strumenti, in particolare JavaScript, noterete una dimensione "dopo esser stato minimizzato e compresso con gzip". Minimizzare un file significa rimuovere gli spazi aggiuntivi, le linee extra e i commenti da un file per ridurne le dimensioni. Un altro modo per ottenere lo stesso risultato è configurare il server per comprimere i file, prima di inviarli all'utente.
Questa configurazione va impostata sul vostro server. Attraverso alcune impostazioni è possibile indicare al server di inviare determinati file compressi per impostazione predefinita. In generale i file HTML, CSS e JavaScript dovrebbero essere compressi. Altri file, come le immagini, lo sono già.

NOTA

Per ulteriori informazioni su come usare la compressione con gzip, c'è un ottimo tutorial su http://rwdwp.com/7.

Siete curiosi di conoscere la vostra velocità di connessione? Farlo è abbastanza facile. Basta seguire questi passaggi!

1. Aprite il vostro browser preferito e puntate su www.speedtest.net
   o
   sul vostro dispositivo mobile scaricate l'app Speedtest.

2. Fate clic sul pulsante Begin Test.
3. Osservate come il servizio:
   a. esegue il ping di un server dalla vostra macchina;
   b. ottiene la velocità di download;
   c. ottiene la velocità di upload.

Quando si tratta di siti web, dovreste essere interessati alle velocità di download. A casa potete raggiungere 30 Mbps (io, a lavoro, raggiungo addirittura 70 Mbps). Una connessione 4G raggiungerà circa 5 Mbps, forse anche 9 in alcune buone giornate. Una connessione 3G, tuttavia, può essere davvero lenta: 1 Mbps o anche meno. Questo è il caso in cui il peso della pagina può davvero influenzare l'utente.
Avremo modo di approfittare di WordPress per questo genere di cose.

## Conclusioni

Con queste parole si conclude con successo il mio primo capitolo, dedicato al Web Design Responsive. Mi auguro che lo abbiate trovato utile, perché ora stiamo per affrontare il motivo principale per cui state leggendo questo libro. Insieme, ci accingiamo a rispondere alla domanda "come posso utilizzare WordPress per fare dei bei siti responsive?".
Da qui in avanti, analizzeremo le tecniche e le implementazioni dei temi di WordPress attraverso le migliori pratiche RWD, utilizzando le funzioni già pronte in WordPress, del JavaScript leggero e alcune istruzioni di rilevamento lato server. L'ultima parte del libro sarà occupata da tutorial stile "libro di ricette" sulle interfacce utente responsive in WordPress.

### Domande

1. Chi ha coniato il termine "Responsive Web Design"?
2. Perché è importante utilizzare i breakpoint basati sugli em rispetto a quelli basati sui px?
3. Qual è il modo migliore per determinare i breakpoint?
4. Quali sono le altre cose che dovete considerare (oltre alle dimensioni dello schermo) quando si crea un sito responsive?

### Risposte

1. Ethan Marcotte ha coniato il termine nel suo articolo dallo stesso titolo su *A List Apart*.
2. I breakpoint basati sugli em sono più flessibili di quelli basati sui px e si adattano alle impostazioni personali dell'utente.

3. Il modo migliore per determinare i breakpoint si basa sul contenuto: si seleziona un breakpoint quando il contenuto inizia a essere visualizzato in modo scorretto.
4. Oltre alle dimensioni dello schermo, è necessario considerare la velocità della connessione, le dimensioni della pagina e il modo più efficace per raggiungere i vostri obiettivi, senza gravare troppo sull'utente.