UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Tecnologie per il Responsive Web Design

Relatore:
Ch.ma Prof.ssa
Maristella Agosti

Laureando:
Renier Alessandro
Matr. 604416

12 Dicembre 2016 ***Anno Accademico 2015-2016***

Indice

Capitolo 1

Introduzione

1.1 Obiettivi

Lo scopo della presente tesi è quello di illustrare in maniera sintetica ma esauriente le tecnologie più recenti e utilizzate nel mondo dello sviluppo di applicazioni Web Responsive. Negli ultimi anni si è verificata una vera e propria esplosione di nuovi linguaggi e framework in grado di rispondere ad una sempre maggiore richiesta di interattività e look&feel da parte degli utenti, oltre alla diffusione capillare di dispositivi mobili con connessione web. Questo ha portato quindi vari team specializzati, tra cui anche grandi colossi come Microsoft e Google, alla creazione di strumenti in grado di aiutare gli sviluppatori nella realizzazione di applicativi web sempre più avanzati.

1.2 Struttura della tesi

Per raggiungere gli obbiettivi della tesi si sono affrontate le tematiche di interesse che sono illustrate nelle seguenti parti della tesi:

- **Il Responsive Web Design (RWD)**: viene introdotto il concetto di RWD, la storia alla base delle necessità che hanno portato alla nascita di tale concetto ed un'illustrazione delle caratteristiche peculiari che pervadono il RWD nelle varie soluzioni che sono scaturite negli ultimi anni a sostegno di tale tipologia di sviluppo.
- **HTML5**: viene spiegato cos'è il linguaggio di markup di ultima generazione, la storia alla base dello sviluppo di tale linguaggio e le caratteristiche tecniche introdotte in quest'ultima versione dell'HTML.
- **Bootstrap**: viene introdotto il pacchetto di layouting Bootstrap, illustrata la storia che ha portato gli sviluppatori di Twitter ad implementare tale pacchetto, le caratteristiche tecniche, oltre alle novità

introdotte con la versione 3. Inoltre viene illustrato lo sviluppo di una semplice pagina responsiva preparata con tale strumento.

- **AngularJS**: viene spiegato cosa è questo framework per lo sviluppo di applicazioni RWD, illustrata la storia della sua comparsa ed analizzate le specifiche tecniche che rendono questo framework il più utilizzato sul fronte dello sviluppo RWD. Infine vengono elencate le sostanziose novità e modifiche tra la versione 1 e la versione 2 di tale strumento.

Le conclusioni poi forniscono un confronto fra i maggiori competitor presenti nel web e una possibile previsione del futuro andamento per quanto riguarda lo sviluppo di applicazioni RWD.

Capitolo 2

Il Responsive Web Design

2.1 Introduzione

Il design responsivo o responsive web design[1] indica una tecnica di web design per la realizzazione di siti web in grado di adattarsi graficamente in modo automatico al dispositivo con i quali vengono visualizzati (ad esempio: monitor di computer con diverse risoluzioni, tablet, smartphone, cellulari, web tv...) riducendo al minimo la necessità dell'utente di ridimensionare e scorrere i contenuti. Si sottolinea che i termini responsive web design, responsive design, design responsivo, adattivo o adattativo saranno usati indifferentemente nella presente tesi.

Il design responsivo è un importante elemento dell'accessibilità, la quale tiene conto di numerosi fattori, incentrati non solo sui dispositivi ma anche sulle caratteristiche dell'utente (quali capacità cognitive, vista, difficoltà fisiche...): si consideri ad esempio il kit di test usato nello sviluppo del sito web della BBC, mostrato in figura 2.1.

[1] la locuzione Responsive Web Design – RWD - è stata coniata da Ethan Marcotte in un articolo sulla rivista A List Apart, descrivendone poi teoria e pratica in un breve saggio del 2011 spesso indicando tale design come RWD

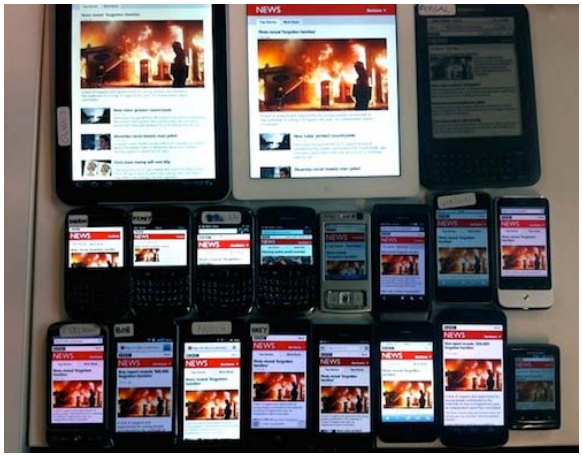

Figura 2.1: *Il kit per i test usato dalla BBC presentato in mobiletestingfordummies.tumblr.com.*

La lingua inglese designa genericamente con l'aggettivo "responsive" tutto ciò che "reagisce o risponde rapidamente e in modo appropriato ad uno stimolo". In italiano l'aggettivo che rende al meglio il termine inglese (avendo "responsivo" un altro significato) è adattativo (o adattivo). Così la definizione dello Zingarelli:

> *(Tecnol.) Capace di adattamento (sistema adattativo: capace di modificarsi per soddisfare nuovi requisiti SIN. Adattivo).*

Se questa idea viene applicata ad un sito web, se cioé quest'ultimo fosse considerato come un "sistema adattivo", si può facilmente ricavare una definizione come quella fornita da Kayla Knight in articolo pubblicato in Smashing Magazine e di seguito riportato in italiano:

> *Con Responsive Design indichiamo quell'approccio per il quale la progettazione e lo sviluppo di un sito dovrebbero adattarsi al comportamento e all'ambiente dell'utente in base a fattori come le dimensioni dello schermo, la piattaforma e l'orientamento del dispositivo.*
> *La pratica consiste in un mix di griglie, layout e immagini flessibili, più un uso accorto delle media query CSS.*
> *Quando l'utente passa dal suo PC desktop ad un iPad, il sito dovrebbe automaticamente adattarsi alla nuova risoluzione, modificare le dimensioni delle immagini e le interazioni basate sugli script. In altre parole, un sito dovrebbe implementare tutte quelle tecnologie utili per un adattamento automatico alle preferenze dell'utente.*

Figura 2.2: *Home page de "Il viaggio" su dispositivi diversi.*

Un efficace confronto è pure quello proposto da Mark Boulton che paragona il responsive design ad un sistema di riscaldamento per la casa. Alla base c'è un termostato: ha dei sensori che misurano la temperatura. Il termostato è anche dotato di un software che possiamo programmare. Ci sono infine dei componenti del sistema che, in base a quanto abbiamo programmato, accendono o spengono il riscaldamento, oppure regolano la temperatura. Nel responsive design i "sensori" sono i browser, il "software" sono i CSS, in particolare le dichiarazioni *@media*, il meccanismo che attiva il sistema nei suoi vari stati sono le dichiarazioni contenute nelle media query.

2.2 Storia

Il design responsivo nasce con la necessità di rendere i siti web facilmente accessibili con ogni tipo dispositivo e risoluzione video, necessità rilevata e studiata sin dagli anni 1990, anche dal gruppo del World Wide Web Consortium. Fino agli ultimi anni, i siti web sono stati progettati in modo che si adattassero bene ai formati più comuni dei schermi desktop e laptop. Nel 2000, ciò significava la progettazione per una larghezza dello schermo di 800 pixel; entro la metà degli anni 2000, la maggior parte dei display era larga 1.024 pixel. Anche se la maggior parte dei monitor disponibili sul mercato erano di pochi formati e dimensioni standard, esistevano monitor in formato wide screen sul mercato, senza contare i formati dei monitor più datati. I Web designer desideravano che i loro siti fossero visualizzati esattamente nel-

la stessa maniera, indipendentemente dal monitor utilizzato, in modo tale da adattare questi siti su ogni formato, come ad esempio i siti a larghezza 960 pixel facilmente adattabili su schermi larghi 1024 pixel. Sugli schermi più ampi sarebbero semplicemente state visualizzate delle barre laterali vuote[2] riempiendo lo spazio supplementare su entrambi i lati del sito come si vede in Figura 2.3.

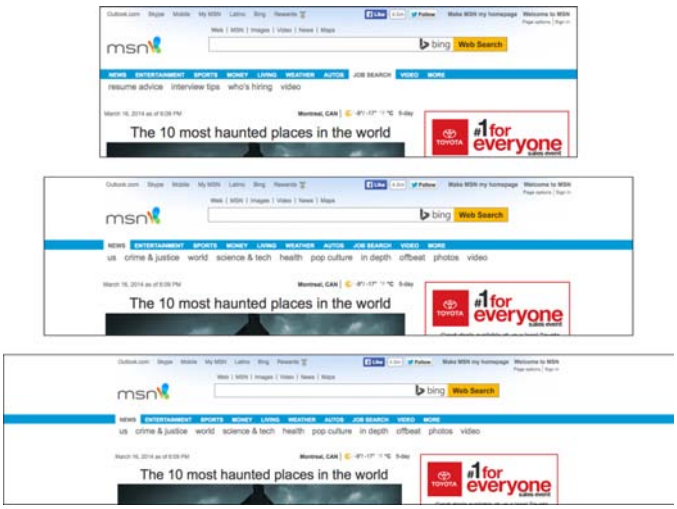

Figura 2.3: *Adattamento del sito www.msn.com su display con larghezze differenti.*

Le idee di *fluid design* e *liquid design* hanno guadagnato un po' di interesse nei primi anni del 2000. Queste tecniche utilizzano larghezze basate su percentuali per consentire la progettazione di una pagina web affinché possa adattarsi alla larghezza dello schermo, in modo da sfruttare lo spazio disponibile sugli schermi più ampi. Sebbene questo fosse un buon metodo per una visualizzazione corretta su monitor di dimensioni differenti, faceva perdere in parte il controllo sul sito progettato, con spiacevoli risultati su monitor di grandi dimensioni, portando la maggior parte dei web designer ad aderire al design a dimensione fissa, più facile da gestire.

2.2.1 Navigazione web Mobile

I primi telefoni cellulari ad avere accesso al web non necessitavano di grandi capacità per visualizzare i contenuti, ed infatti si limitavano a visualizzare i dati solo in forma testuale, come previsioni meteo o risultati sportivi, visto che il primo browser mobile poteva visualizzare solo HTML di base, spesso

[2]queste barre laterali sono chiamate *whitespace* in termini tecnici

in scala di grigi. E' stato solo a metà degli anni 2000 che i browser mobile furono in grado di visualizzare pagine web "reali" usando tecnologie quali CSS2 e JavaScript, su dispositivi avanzati come gli smartphone. Il punto di svolta è stato appunto l'iPhone, uscito nel 2007, un dispositivo sufficientemente potente da poter usare le tecnologie degli ultimi browser mobile, oltre alla possibilità, grazie allo schermo multi-touch, di visualizzare una pagina web completa ed effettuare lo zoom sulla zona di interesse: questa è stata la soluzione adottata da Apple in quanto i siti web erano progettati per essere visualizzati su monitor di 960 pixel, mentre lo schermo dell'iPhone era di soli 320 pixel.
Tale stratagemma era però inaccettabile per una buona user experience, costringendo i web designer a cercare un nuovo modo per rendere le pagine web maggiormente fruibili su schermi di piccole dimensioni: nacque in questo modo il concetto di siti web per cellulari. Questi siti sono una versione specifica e separata per la loro fruizione da mobile[3], progettati appositamente per schermi larghi 320 pixel: gli utenti erano spesso reindirizzati automaticamente alla versione mobile del sito se stavano usando un telefono cellulare. Questa pratica ha portato questi siti mobili separati ad essere noti come siti web *m-dot*. Naturalmente, questo sdoppiamento dei contenuti su siti distinti (nonostante la versione mobile fosse spesso un copia ridotta del sito ufficiale come mostrato in Figura 2.4) comportava un lavoro extra ai web designer, che era accettabile solo agli albori del mercato smartphone.

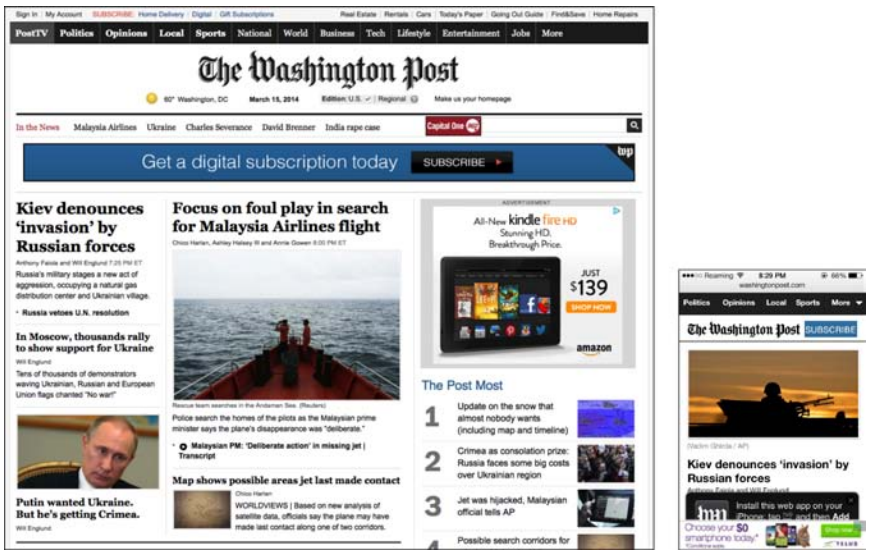

Figura 2.4: *Versioni desktop e mobile del sito www.washingtonpost.com.*

Infatti, pochi anni dopo l'introduzione dell'iPhone, che di fatto rendeva

[3]vengono contraddistinti da URL con un sottodominio con prefisso m. - ad esempio *http://m.sprint.com/*

Apple l'unico competitor sullo scenario del mobile web, altri smartphone apparvero sul mercato, con schermi di dimensioni e formati molto diversi tra loro: i siti *m-dot*, progettati specificatamente per schermi larghi 320 pixel, non erano più adatti alla navigazione web da mobile. Poi, nel 2010, Apple ha rilasciato l'iPad. Ancora una volta, questo è stato un punto di svolta. I siti *m-dot* erano troppo piccoli per approfittare dello schermo molto più ampio dell'iPad, mentre i siti a larghezza fissa per il desktop erano comunque troppo grandi per essere facilmente visualizzati in modalità verticale su un iPad. Alcuni progettisti realizzarono quindi siti web specifici per l'iPad (portando ad avere ben tre siti separati), ma con l'entrata nel mercato tablet di altre aziende non era più sostenibile creare siti web ad hoc per ogni possibile dimensione dello schermo.

2.2.2 Media Query

La comunità del web design ancora una volta tornò al concetto di layout fluido, utilizzando larghezze basate su percentuali, cercando di capire come trovare una soluzione per i dispositivi più piccoli. Utilizzare le percentuali invece dei pixel consente a una pagina web e relative sezioni di modificare la larghezza per adattarsi a qualsiasi dimensione, quindi diminuendo le differenze tra le varie visualizzazioni di dispositivi della stessa fascia, ma ancora una volta tra dispositivi con schermi significativamente differenti, il disallineamento delle visualizzazioni era evidente. In sostanza, il problema era questo: senza dover creare siti separati, come può un sito web essere visualizzato in una colonna sugli schermi stretti e su colonne multiple su schermi più ampi? Come si può chiedere al browser di apportare modifiche al progetto, sulla base delle caratteristiche del dispositivo su cui si sta visualizzando il sito? La risposta è tramite le *media query*. La regola CSS *@media*, che permette di visualizzare diversi stili CSS sulla base delle caratteristiche dei dispositivi, era in realtà parte di CSS2 più di un decennio fa, ma allora supportava solo funzionalità per la stampa di un sito web (che può comprendere modifiche base, come la rimozione dei colori di sfondo che evitavano di sprecare inchiostro della stampante), ma la sua utilità non andò oltre. Solo a partire dai CSS3 le specifiche (quindi una formale e dettagliata descrizione di come erano definite) per le media query hanno funzionalità più precise basate su un supporto (dispositivo) con determinate caratteristiche, come ad esempio larghezza, altezza, colore e capacità. Le media query non influiscono sulla struttura HTML e riguardano solo gli stili applicati alla pagina usando i CSS (i browser hanno iniziato a supportare le media query CSS3 verso il 2009), quindi è possibile dare un diverso layout al sito web in base alle dimensioni dello schermo su cui viene visualizzato, senza dover creare siti web separati.

2.2.3 Flessibilità

Le media query possono riorganizzare il layout, ma il responsive design non funzionerebbe senza un fondamento di flessibilità: ogni misura orizzontale su un sito ha bisogno di essere in unità flessibili anziché pixel inflessibili. Questo significa che la larghezza delle colonne e altri elementi del layout sarà in percentuale, e il testo sarà misurato in una unità relativa chiamata *em*. Il dimensionamento di immagini sulla pagina funziona un po' diversamente, perché non vogliono necessariamente cambiare le dimensioni a seconda della larghezza dello schermo: il problema è che in base alle dimensioni dello schermo del dispositivo, non sempre può esserci abbastanza spazio per visualizzare una immagine in modalità full size. È necessario fare in modo che l'immagine non venga tagliata fuori se non si adatta.

2.2.4 Responsive Web Design

Nessuna di queste idee, media query o la flessibilità, era nuovo o innovativo di per sé. Ma nel 2010, il web designer Ethan Marcotte ha trovato un modo per utilizzare questi concetti insieme per sviluppare siti web che avrebbero risposto alle diverse dimensioni dello schermo: tramite il Responsive Web Design.

2.3 Perché Responsive Design

Il concetto di responsive design è stato dibattuto da quando è stato introdotto. La ragione più convincente per l'utilizzo del design adattativo è che la creazione di un sito web non solo visivamente piacevole ma che funzioni correttamente sui dispositivi che si trovano sul mercato in un determinato momento, è che questo tipo di sviluppo permetterà il corretto funzionamento e la corretta visualizzazione dei contenuti anche su dispositivi che saranno disponibili in futuro. Inoltre, con il responsive design non si corre il rischio che gli utenti visualizzino la versione mobile di un sito su un monitor desktop, o vice versa. Se si dispone di siti web separati, nel caso si utilizzi il rilevamento del dispositivo per inviare la versione corrispondente del sito per ogni dispositivo o se si utilizza un set separato di URL (come un sottodominio *m-dot*) per servire un sito mobile, può portare ad un rendering errato dei contenuti. Siti che hanno una versione mobile separata usano comunemente il rilevamento dei dispositivi per determinare quale versione di una pagina web (desktop o mobile) si dovrebbe inviare a qualsiasi dispositivo particolare. In questo modo, ogni pagina del sito potrà avere un solo URL, anche se ci sono in realtà due versioni separate con diversi codici HTML. Tuttavia, questo processo non è accurato al 100%, e a volte sarà inviata la versione non corretta della pagina. Inoltre, il processo di rilevamento del dispositivo

può aumentare il tempo di caricamento della pagina. Con collegamenti che vengono scambiati tra gli utenti attraverso i social media o e-mail, ottenere la versione corretta di una pagina sarà spesso un compito supplementare per gli utenti, senza contare i casi in cui non avranno affatto questa opzione. Ad esempio, se un utente del desktop inviasse una e-mail con un link dal "The New York Times" ad un utente mobile, l'utente mobile otterrà un messaggio nella parte superiore dello schermo per fargli capire che c'è una versione mobile del sito, come in Figura 2.5.

Figura 2.5: *Il sito http://www.nytimes.com/ con opzione di scelta della versione.*

Piacevole, ma richiede lavoro aggiuntivo da parte del utente, e tempo supplementare per scegliere e quindi caricare una pagina completamente separata. D'altra parte, se sto visitando il sito mobile del New York Times dal mio telefono e invio una mail di un link ad un articolo a qualcuno che apre il link su un computer desktop, si ottiene quello che si vede in Figura 2.6: una pagina ottimizzata per cellulari, e non c'è un modo chiaro per andare al sito web desktop completo.

Figura 2.6: *Il sito mobile http://mobile.nytimes.com/ visualizzato su desktop.*

L'utente può leggere l'articolo, ma sarà suo dovere fare clic per vedere le versioni *full-size* di tutte le immagini, oltre a visualizzare tutti i collegamenti supplementari e articoli che sono presenti sulla versione desktop del sito consigliato. Con un design reattivo, si dispone di una sola pagina web, in modo da non visualizzare mai correttamente la "versione sbagliata". Al sito non importa su quale dispositivo è in fase di visualizzazione.

2.3.1 Meno lavoro

Il vantaggio più evidente di utilizzare il design reattivo è che per creare un sito web, un disegno, un set di codice o di contenuti, è necessario un unico progetto. Se si dispone di una versione separata solo per cellulari del sito, si dovrà creare e mantenere due (o più) gruppi completamente separati di istruzioni HTML. Le modifiche dovranno essere effettuate su ciascun sito, e anche se si sta cercando di tenerli allineati, ci saranno quasi certamente problemi e qualcosa finirà per non corrispondere tra le versioni. Sebbene l'utilizzo di un Content Management System (CMS) o un sistema di template possano rendere il lavoro più facile, non vi sono maggiori contenuti o codice da mantenere, o altri componenti che si possono potenzialmente "rompere". Con un sito reattivo, si ha un solo set di contenuti, e sarà visualizzato in modo appropriato, non importa quale sia la dimensione dello schermo: future modifiche di progettazione possono essere effettuate tramite interventi mirati apportati al foglio di stile. Per gli sviluppatori inesperti sul responsive design l'iniziale compito di creare un sito web reattivo può richiedere uno sforzo maggiore rispetto alla creazione di un sito a dimensione fissa, ma nel lungo termine si avrà meno lavoro da fare per gestire il sito.

2.3.2 Ottimizzato per il Search

Un sito mobile separato, con una serie separata di URL, può inoltre creare problemi con il posizionamento del sito nei motori di ricerca. Se si dispone di due versioni separate di una pagina con lo stesso contenuto o simile, ma URL diversi (ad esempio *http://www.example.com/* e *http: // m.example.com/*), i motori di ricerca hanno bisogno di sapere che essi sono considerati la stessa pagina in modo che questa possa essere indicizzata correttamente e visualizzata come un'unica voce nella lista dei risultati di ricerca. Anche se questo è possibile utilizzando JavaScript o il codice sul server, è un po' complicato, e se non si riesce a farlo correttamente si può finire con la visualizzazione di entrambe le versioni di una pagina nei risultati della ricerca, confondendo gli utenti, oltre ad influenzare negativamente il ranking di ricerca. Google ha raccomandato il responsive design ottimizzato per smartphone dal 2012, non solo perché crea una migliore user experience, ma anche perché permette al crawler di Google di recuperare i risultati in modo più efficiente, il che significa che le modifiche al sito probabilmente possono essere aggiornate nei risultati di ricerca in modo più rapido.

2.4 Caratteristiche del RWD

Un sito responsivo fa uso di una impaginazione grafica con griglie a proporzioni fluide, struttura e immagini flessibili, e, generalmente, dei fogli di stile CSS3; in particolare di un'estensione della regola *@media*, per adattare l'impaginazione grafica all'ambiente nel quale il sito è visualizzato:

- Le media query consentono alla pagina di usare diversi fogli di stile in base alle caratteristiche del dispositivo utilizzato
- Il concetto di griglia flessibile richiede che gli elementi siano dimensionati tramite unità relative come percentuali ed em, e non con unità assolute come pixel o punti
- Le immagini flessibili devono poter essere visualizzate con dimensioni diverse, in modo da potersi adattare all'impaginazione evitando di sovrapporsi agli altri elementi.

Come risultato, gli utenti che utilizzano diverse periferiche e browser, hanno accesso a un singolo sorgente i cui contenuti vengono però disposti in modo differente e tale da essere sempre facilmente consultabile, e senza dover compiere troppe operazioni di ridimensionamento, scorrimento e spostamento.

2.4.1 Resolution breakpoint

La necessità di adattare l'impaginazione alle diverse dimensioni e risoluzioni degli schermi ha introdotto il concetto di *"Resolution breakpoint"* ("punti di interruzione della risoluzione"), in modo da stabilire delle soglie alle quali modificare la presentazione grafica in funzione della larghezza del dispositivo. Tali soglie sono generalmente espresse in pixel, anche se l'aumento della densità dei pixel nelle nuove generazioni di dispositivi comporta che non si possa considerare l'area di visualizzazione solo in termini di pixel, senza considerarne l'effettiva dimensione. Il framework Bootstrap identifica (in riferimento al *"max-device-width"*) quattro tipi di device e corrispondenti resolution breakpoint:

- *extra small device* con risoluzione inferiore a 768 pixel
- *small device* con risoluzione fino a 992 pixel
- *desktop* con risoluzione inferiore a 1200 pixel
- *large device* con risoluzione superiore a 1200 pixel.

Mentre Ethan Marcotte ne identifica sei:

- 320 pixel per dispositivi con schermi piccoli, come cellulari, con orientamento verticale (*portrait*)
- 480 pixel per dispositivi con schermi piccoli, come cellulari, con orientamento orizzontale (*landscape*)
- 600 pixel piccoli tablet, come Kindle di Amazon (600x800) e Nook di BarnesNoble (600x1024), con orientamento verticale
- 768 pixel tablet da 10 pollici, come l'iPad (768x1024), con orientamento verticale
- 1024 pixel computer da scrivania, fissi o portatili e tablet come l'iPad (1024x768), con orientamento orizzontale
- 1200 pixel computer con schermi larghi, tipicamente fissi ma anche alcuni portatili.

Tali tipologie possono essere più genericamente ricondotte a quattro principali:

- *Mobile*: per cellulari
- *Narrow*: per tablet
- *Normal*: computer fisso o portatile
- *Wide*: schermi di grandi dimensioni.

2.4.2 "Pattern" per il responsive design

Intorno a queste idee sta crescendo non soltanto l'interesse degli sviluppatori ma anche la volontà di iniziare a fissare un insieme di *best practice*, individuando per esempio i principali pattern in fatto di design, che si tratti di menu di navigazione o di layout. Esiste anche un sito che è proposto come modello e riferimento, vista la sua complessità: quello del Boston Globe. Molti hanno considerato la presentazione di questo sito una sorta di pietra miliare, paragonando l'evento al rilascio dei primi siti "importanti" realizzati con layout CSS invece che con le tabelle. Quei siti furono la prova che, dopo la fase pionieristica in cui la nuova idea veniva applicata per lo più a progetti personali o di limitato impatto in termini di traffico sul web, ci si poteva spingere a implementarli in contesti ben più ampi.

2.4.3 Responsive design nella pratica

Da quanto si è detto finora, si potrebbe ricavare l'idea che il responsive design abbia a che fare essenzialmente con i CSS o con l'adattamento del layout. Certo, il meccanismo che oggi è l'architrave di questo approccio sono le media query CSS. E se si tratta di costruire layout che si adattano, è sempre con i CSS che si opera. Ma, naturalmente, al crescere della complessità del progetto, aumenta la necessità di coinvolgere linguaggi come Javascript, tecnologie come Ajax (jQuery), interazioni lato server. Quello che si vuole rendere chiaro, però, è che a prescindere dai mezzi è la filosofia di fondo che conta. Di base sono questi gli obiettivi minimi da porsi per realizzare un'esperienza utente positiva nei contesti d'uso più diversi:

- Adattare il layout al più ampio numero di risoluzioni di schermo possibile (dai telefoni cellulari al desktop)
- Adattare le dimensioni delle immagini (e in genere di tutti i contenuti a larghezza fissa) alla risoluzione e alle dimensioni dello schermo;
- Servire immagini "meno pesanti" ai dispositivi che non possono sempre sfruttare la banda larga
- Semplificare il layout degli elementi presenti sulla pagina per i dispositivi mobili con schermi piccoli
- Nascondere gli elementi non essenziali su questi stessi dispositivi
- Fornire un'interfaccia adatta all'interazione touch per i device che la supportano
- Individuare e sfruttare, quando serve, funzionalità adatte al mobile (come la geolocalizzazione).

2.4.4 Impaginazione grafica

Le strategie per riorganizzare i contenuti in funzione dei dispositivi, hanno portato alla classificazione di diverse tipologie di impaginazioni grafiche :

- **Reflowing**
- **Equal Width**
- **Off Canvas**
- **Source-Order Shift**
- **List**
- **Grid Block.**

L'impaginazione di tipo *Reflowing*, visibile in Figura 2.7, contiene diverse sottocategorie: *Mostly Fluid* (multi colonna con margini più larghi su grandi schermi, e su schermi *narrow* le aree vengono allineate su un'unica colonna), *Column Drop*, *Layout Shifter*, *Tiny Tweaks*. L'impaginazione di tipo *Equal Width* divide lo schermo in colonne delle stesse dimensioni, e il numero di colonne può aumentare o diminuire proporzionalmente alla larghezza dello schermo. L'impaginazione di tipo *Off Canvas* divide lo schermo in aree, principalmente verticali, che al diminuire della risoluzione non vengono mostrate in funzione della loro importanza fino a mostrare una sola colonna con il contenuto principale. L'impaginazione di tipo *Source-Order Shift* sfrutta le proprietà *flexbox* e *box-ordinal-group* dei CSS per cambiare l'ordine con i quali i blocchi di contenuti vengono visualizzati nella pagina. L'impaginazione di tipo *List* organizza la pagina in semplici liste di elementi che, analogamente a quanto succede sulle impaginazioni di tipo *Equal Width*, sono visualizzate su un numero di colonne proporzionali alla larghezza dello schermo così come le impaginazioni di tipo *Grid Block*

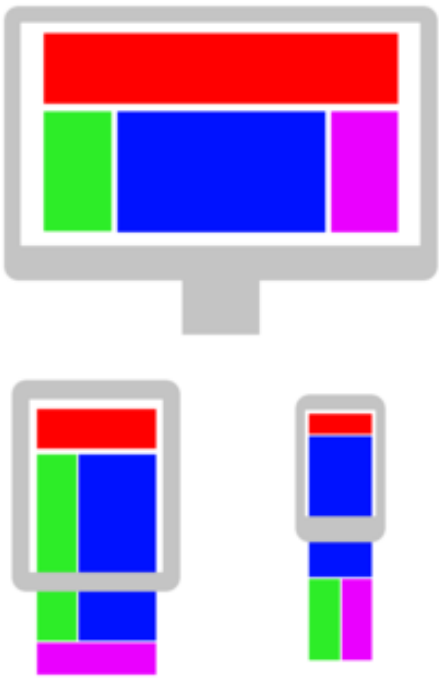

Figura 2.7: *Un esempio di riorganizzazione con layout reflowing dei contenuti su device desktop, tablet e mobile.*

che suddividono il layout in una griglia di rettangoli o quadrati. Leggendo in rete qualche articolo o tutorial dedicato al responsive design, ci si può imbattere facilmente in due espressioni: *content first* (prima i contenuti) e *mobile first* (prima il mobile).

2.4.5 Gestione dei contenuti

Qualunque sia il livello del progetto, è opportuno utilizzare sin dall'inizio un approccio che metta al centro la gestione dei contenuti. Come detto da Jeremy Keith in un articolo del 2012, di seguito riportato in italiano:

> *Non iniziate la progettazione del sito pensando al layout per il desktop. Ma non iniziate nemmeno pensando al layout per il mobile. Pensate invece, prima di tutto, ai contenuti.*

Per "contenuti" non si intendono qui semplicemente i testi, le immagini e gli altri elementi multimediali che costituiscono la base informativa del sito. Sono "contenuti" anche il logo, il menu di navigazione principale, le sezioni secondarie, i moduli per la ricerca sul sito, le inserzioni pubblicitarie, etc. Non è necessario in questa fase iniziale alzare il livello di complessità ricorrendo a strumenti particolarmente sofisticati. In progetti di piccole dimensioni sarà sufficiente affidarsi ad un foglio di carta e a una penna (un esempio in Figura 2.8) per tracciare liste degli elementi che costituiscono i nostri contenuti e per iniziare a buttare giù uno schizzo grossolano dei contenuti stessi nel contesto delle dimensioni di layout che intendiamo supportare.

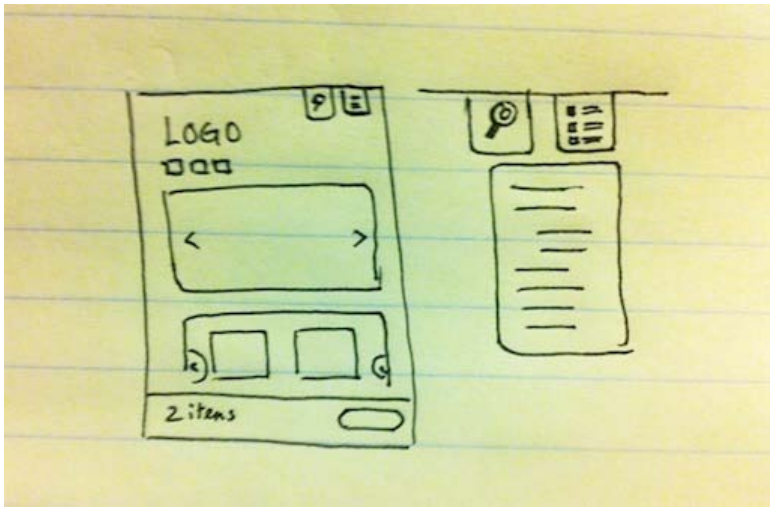

Figura 2.8: *Usare carta e penna.*

Fondamentale è porsi le domande giuste in fatto di strategia per la gestione dei contenuti, passando dal "cosa" (quali contenuti) al "come" (tecniche e modalità di presentazione degli stessi):

- Quali contenuti inseriamo sulla pagina?
- Per i dispositivi mobili inseriamo gli stessi contenuti e le stesse funzionalità che usiamo per il desktop?
- Se non inseriamo gli stessi contenuti, quali manteniamo?
- In che ordine li collochiamo? questo elemento va "sotto" o "sopra" a quest'altro?
- Sui dispositivi mobili come trattiamo i contenuti "superflui"? li nascondiamo? li compattiamo?

Un buon suggerimento rispetto a questa strategia viene da Tyler Herman che in un articolo spiega il proprio punto di vista, di seguito riportato in italiano:

> *Personalmente tendo ad utilizzare un approccio basato su priorità. Basta fare una lista di tutti i contenuti che dovranno essere presentati sulla home page. Poi si ordina la lista, partendo dagli elementi più importanti fino ai meno importanti. Infine, si prende quella stessa lista e si riordinano gli elementi dal più al meno importante per ogni breakpoint. Ogni layout ha le sue priorità e le sue specifiche considerazioni da valutare in fatto di design.*

L'esempio classico che accompagna questo tipo di considerazione è quello del sito di un ristorante. Sul desktop potrò offrire una sontuosa rappresentazione grafica delle varie portate del menu, dando dunque enfasi e centralità a foto o animazioni. Sul mobile si può presumere che chi usa il sito andrà prioritariamente in cerca di informazioni di contatto, magari per fare una prenotazione. E potrebbe essere anche utile inserire il rating delle recensioni da siti come *Trip Advisor*. Invece che immagini da 800kb dei primi e dei secondi, potrei inserire un elenco testuale dei piatti, evidenziando i prezzi. Tutto, insomma, a vantaggio di un utente che opera con un terminale meno capace (ad esempio schermo piccolo, banda non necessariamente larga...) e che in mobilità ha necessità di informazioni veloci e adatte alle esigenze del momento.

2.4.6 Partire dal mobile

La paternità del concetto di *mobile first* spetta a Luke Wroblewski. Secondo questo approccio, nella progettazione del sito, è necessario partire dalla

strutturazione dei contenuti per lo schermo limitato nelle dimensioni dei dispositivi mobili. Non si parte, quindi, da un layout e da un set di funzionalità concepiti per il desktop per poi adattarli per il mobile. Al contrario, si parte dai limiti posti dai dispositivi mobili per concentrarsi al meglio sui contenuti essenziali, per poi arricchire il tutto sui dispositivi più capaci, man mano che insomma aumentano le dimensioni dello schermo (ma anche la potenza dell'hardware). In sintesi: l'approccio del responsive design non è altro che una versione rivista e corretta del cosiddetto *progressive enhancement*.

2.4.7 Rilevazione del dispositivo e compatibilità

Un tema di particolare interesse per il design responsivo è quello della compatibilità. Non tutti i browser e dispositivi, infatti, riconoscono le istruzioni più utilizzate per il ridimensionamento fluido dei contenuti o supportano le tecnologie necessarie. È inoltre fondamentale la rilevazione corretta del dispositivo, per fornire la relativa impaginazione grafica e individuare il livello di compatibilità possibile. I browser dei primi cellulari non sono in grado di interpretare funzioni quali media query o JavaScript, ed è pertanto più conveniente creare una impaginazione specificamente adatta alla visualizzazione su smartphone e computer, piuttosto che tentare una "degradazione graduale" per adattare un sito complesso e carico di immagini alla maggior parte dei cellulari. L'identificazione dello user agent, ovvero del browser, e l'identificazione del dispositivo mobile sono due modi di dedurre se JavaScript e alcune istruzioni dell'HTML e dei fogli di stile sono supportate. L'utilizzo di librerie JavaScript come Modernizr, jQuery, e jQuery mobile può essere utile allo scopo, verificando direttamente le caratteristiche e lo user agent usati dall'utente.

Capitolo 3

HTML 5

3.1 Introduzione

Si consideri innanzitutto la definizione di HTML5:

> *HTML5 è un linguaggio di markup per la strutturazione delle pagine web, pubblicato come W3C Recommendation ad ottobre 2014.*

Si consideri quindi cosa sono i linguaggi di markup:

> *In generale un linguaggio di markup è un insieme di regole che descrivono i meccanismi di rappresentazione (strutturali, semantici o presentazionali) di un testo che, utilizzando convenzioni standardizzate, sono utilizzabili su più supporti. La tecnica di formattazione per mezzo di marcatori (o espressioni codificate) richiede quindi una serie di convenzioni, ovvero un linguaggio a marcatori di documenti.*

Il panorama di Internet è cambiato molto a seguito dell'assunzione a *W3C Reccomandation* della versione precedente delle specifiche, avvenuta verso la fine del 1999. In questo periodo il Web era strettamente legato al concetto di ipertesto e l'azione più comune per l'utente era la fruizione di contenuti, tipicamente in forma testuale. La mediamente bassa velocità di connessione e il limitato investimento sui media contribuivano ad una scarsa presenza di applicazioni web, più costose da sviluppare ed esigenti in termini di banda. Tutto questo era ben rappresentato da un linguaggio, HTML, principalmente orientato ad agevolare la stesura di semplici documenti testuali collegati fra loro. Negli anni successivi l'interesse intorno alla rete ha subìto una brusca accelerazione e questo ha condizionato positivamente sia la diffusione che la velocità di connessione della stessa, attirando di conseguenza maggiori investimenti e ricerca. Al modello di fruizione dei contenuti si è aggiunta la possibilità per l'utente finale di divenire esso stesso creatore attraverso applicazioni web sempre più elaborate ed interessanti. Questo nuovo livello di

complessità, in termini di sviluppo, ha però dovuto scontrarsi con un set di specifiche poco inclini ad essere utilizzate per tali fini e che quindi si sono prestate al compito solo a scapito di infiniti workaround. Parallelamente il percorso di crescita del web ha fatto emergere alcune strutture di contenuto ricorrenti, ben caratterizzate dal fenomeno dei blog: informazioni di testata, menu di navigazione, elenchi di articoli, testo a pié di pagina, ed altri. La parte dedicata al singolo articolo presenta anch'essa solitamente lo stesso set di informazioni quali autore, data di pubblicazione, titolo e corpo del messaggio. Anche in questo caso il linguaggio HTML4 non ha saputo fornire gli strumenti adatti a consentire una corretta gestione e classificazione del contenuto obbligando gli sviluppatori web a ripiegare su strutture anonime, quali * * e *<p>*, arricchite di valore semantico con l'utilizzo di attributi quali *class* e *id*. HTML5 nasce per risolvere questi problemi offrendo agli sviluppatori web un linguaggio pronto ad essere plasmato secondo le più recenti necessità, sia dal lato della strutturazione del contenuto che da quello dello sviluppo di vere e proprie applicazioni. Quindi HTML5 si presenta non solo come una revisione del linguaggio, ma anche come un tool di sviluppo comprendente un vastissimo elenco di funzionalità che si pongono attorno al tema del linguaggio di markup; per prima cosa è importante ricordare che, anche in virtù della storia della sua nascita, all'interno di HTML5 convivono in armonia due anime: la prima, che raccoglie l'eredità semantica di XHTML2, e la seconda che invece deriva dallo sforzo di aiutare lo sviluppo di applicazioni Web. Il risultato di questo mix di intenti è più articolato di quanto si possa immaginare; in prima istanza si assiste ad una evoluzione del modello di markup, che non solo si amplia per accogliere nuovi elementi, ma modifica in modo sensibile anche le basi della propria sintassi e le regole per la disposizione dei contenuti sulla pagina. A questo segue un rinvigorimento delle API JavaScript che vengono estese per supportare tutte le funzionalità di cui una applicazione moderna potrebbe aver bisogno:

- Salvare informazioni sul device dell'utente
- Accedere all'applicazione anche in assenza di una connessione Web
- Comunicare in modo bidirezionale sia con il server sia con altre applicazioni
- Eseguire operazioni in background
- Pilotare flussi multimediali (video, audio)
- Editare contenuti anche complessi, come documenti di testo
- Pilotare lo storico della navigazione
- Usare metafore di interazione tipiche di applicazioni desktop, come il drag and drop

- Generare grafica 2D in tempo reale
- Generare grafica 3D in tempo reale
- Accedere e manipolare informazioni generate in tempo reale dall'utente attraverso sensori multimediali quali microfono e webcam.

Attorno a quello che può essere definito il nucleo autentico delle specifiche gravitano tutta una serie di altre iniziative, alcune delle quali in avanzato stato di definizione, studiate per:

- Accedere alle informazioni geografiche del device dell'utente (ad esempio posizione, orientamento...)
- Mantenere un database sul device dell'utente
- Generare grafica 3D in tempo reale.

E non si dimentichi che l'evoluzione del linguaggio HTML viaggia di pari passo con quella dei fogli di stile CSS, arrivati alla terza versione, e di altri importanti standard come SVG e MathML, e che ognuno di questi componenti è progettato nella sua versione più recente per recare e ricevere beneficio dagli altri.

3.2 Storia

Per capire veramente ciò che oggi è HTML5 bisogna conoscere la strada intrapresa quando nacque la prima versione di HTML nel 1989, mentre si proponeva sul web come un formato piuttosto povero rispetto ad altri sistemi di marcatura ipertestuale: infatti in questa versione si potevano aggiungere collegamenti ipertestuali (*link*) ad un documento, senza però conoscere la presenza di altri documenti che rimandavano ad esso. Nel 1994 ogni browser aggiungeva le proprie estensioni ad HTML, ad esempio caratteri lampeggianti o gif animate, ma spesso queste estensioni non avevano compatibilità *cross-browser*. Per introdurre uno standard da seguire per gli sviluppatori di web browser nacque il *World Wide Web Consortium*: il W3C. Grazie alla fondazione di tale organizzazione internazionale è stato possibile sviluppare tutte le potenzialità del World Wide Web. Da quel momento si possono definire come milestones i seguenti anni:

- 1997: HTML 3.2 – viene incluso Javascript
- 1999: HTML 4.0
- 2000: XHTML 1.0 – HTML4 ma secondo le regole di XML.

Ma è stato nel 2004 che si ebbe una significativa svolta nella definizione del linguaggio HTML: Ian Hickson annuncia la creazione del gruppo di ricerca WHAT (*Web Hypertext Application Technology*), con il preciso scopo di elaborare specifiche per lo sviluppo di un web più orientato alle applicazioni che ai documenti; tra i sottoscrittori di questa iniziativa si annoverano aziende del calibro di Apple, Mozilla e Opera. Diversamente decise di fare il W3C, che era maggiormente orientato ai concetti di accessibilità, semantica e all'abbandono di HTML in favore di XHTML 2. Questo piccolo scisma dal W3C è determinato dal disaccordo in merito alla rotta decisa dal consorzio durante un famoso convegno del 2004 dove, per un pugno di voti, prevalse la linea orientata ai documenti di XHTML2. *"XHTML 2.0 considered harmful"* è il titolo di un messaggio inviato alla mailing list ufficiale del W3C datato 14 gennaio 2003 che ben riassume i sentimenti contrastanti che all'epoca si respiravano in merito a queste specifiche. La principale causa di perplessità è da ricercarsi nella decisione azzardata di non mantenere la retro-compatibilità con la versione 1.1 eliminando alcuni tag e imponendo agli sviluppatori web un controllo rigoroso nella creazione delle pagine, pena lo stop del processo di parsing e la visualizzazione a schermo degli errori di interpretazione. Nei due anni successivi i gruppi XHTML2 e WHAT proseguono i lavori sulle proprie specifiche in modo indipendente e parallelo, sollevando dubbi e confusione sia da parte dei produttori di browser che degli sviluppatori web. Emblematico in tal senso è un articolo firmato da Edd Dumbill nel dicembre 2005 intitolato *The future of HTML*. Il 27 ottobre 2006 in un post sul proprio blog intitolato *Reinventing HTML*, di seguito riportato in italiano, Tim Berners-Lee annuncia la volontà di creare un nuovo gruppo di ricerca che strizzi l'occhio al WHAT e ammette alcuni sbagli commessi seguendo la filosofia XHTML2:

> *Dopo diversi anni, con il senno di poi, alcune cose sono più chiare.È necessario far evolvere HTML in maniera incrementale. Il tentativo di costringere il mondo a passare a XML in una sola volta, inclusa l'aggiunta di virgolette sui valori degli attributi, le barre in tag vuoti e namespace, non ha funzionato. Il largo pubblico dell'HTML nativo non ha cambiato procedura di sviluppo, soprattutto a causa del mancato funzionamento sui browser. Alcune grandi comunità hanno effettuato tale cambiamento, godendo dei frutti dei sistemi ben formati, ma non tutte. È importante mantenere HTML incrementale, nonché continuare una transizione ai sistemi ben formati, e sviluppando maggiore potenza in questi sistemi.Tim Berners-Lee.*

Dovranno passare quasi altri due anni di competizione tra le due specifiche, questa volta entrambe interne al W3C, prima che nel luglio del 2009 lo stesso Tim sancisca di non voler riconfermare il gruppo XHTML2 per l'anno successivo preferendo di fatto la direzione intrapresa dall'HTML5. Frattan-

to il gruppo di ricerca, formato da una commistione di elementi del W3C e del WHAT, sotto la guida di Ian Hickson, è giunto alla 4 versione delle specifiche. Nonostante il continuo e solerte lavoro e il grande intervallo di tempo speso nella stesura di questo documento, i passi residui necessari ad eleggere questa tecnologia al rango di *'W3C Reccomandation'*, relegandola così tra gli standard riconosciuti, sono ancora molti, al punto che si prevede di raggiungere l'agognato traguardo soltanto attorno al 2020. Si ricorda però che il consorzio si riferisce sempre all'intero universo inscritto nelle specifiche mentre alcune feature ritenute peculiari ed importanti, ad esempio il tag *<video>*, sono già implementate da tempo dalla totalità (o quasi) dei browser in commercio. Nelle immagini seguenti due screenshot del sito del New York Times, rispettivamente nel 1996 e nel 2016.

Si noti il tipico standard delle pagine HTML degli anni 90: un menù statico sulla sinistra e il contenuto mostrato nel riquadro principale, con la presenza di semplici link ipertestuali per accedere ad altre pagine (statiche) contenenti il testo della notizia ed eventualmente una o più immagini.

Figura 3.1: *il sito del New York Times nel 1996*

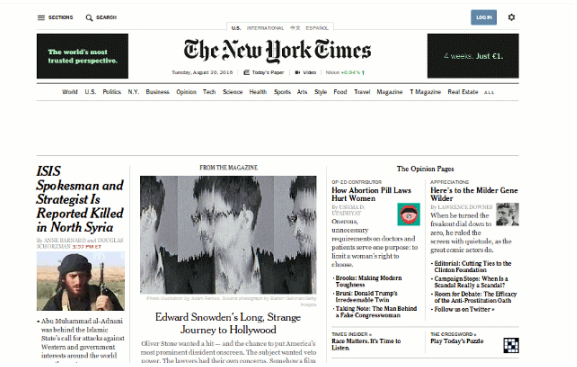

Figura 3.2: *il sito del New York Times nel 2016*

In questa pagina si noti innanzitutto la formattazione adatta a schermi più ampi, menù interattivi ed eventuali animazioni: la lettura delle notizie non avviene più in maniera "passiva" ma l'utente interagisce con la pagina anche tramite l'utilizzo di applicazioni quali il cruciverba del New York Times o la visione di video che possono essere condivisi tramite i social network quali Facebook, Twitter o Pinterest.

3.3 Specifiche

Si passa ora ad illustrare le tecnologie che compongono HTML5: questo infatti non è solo un linguaggio di markup, ma un ambiente, con almeno tre linguaggi:

- HTML v.5
- CSS v.3
- Javascript.

Figura 3.3: *I componenti per lo sviluppo ottimale in HTML5.*

Si può parlare quindi di un ecosistema fortemente orientato a fornire servizi web, nato con una consapevolezza discreta per il futuro del web, una astrazione delle volontà e delle esigenze degli utenti, tenendo ben conto delle frustrazioni da tecnologia che hanno funestato Internet negli ultimi 10/15 anni. Questo ne fa un framework di lavoro che induce a operare seguendo alcuni precisi pattern di design:

- **Degrade Gracefully** : un contenuto Web deve "degradare con grazia" anche verso user agents come i web browser più vecchi o limitati (nei limiti del possibile)
- **Separation of Concerns** (SoC):
  - HTML per i contenuti

  - CSS per la loro presentazione
  - JS per le interazioni della pagina
  - Classi di stile e classi funzionali

- **Interoperability**: compatibilità dei contenuti
- **Universal Access**: compatibilità dei contenitori (computer desktop, tablet o smartphone).

Inoltre HTML5 è stato pensato per semplificare il suo utilizzo: ne sono importanti esempi l'utilizzo di una doctype semplice e chiara e il set di meta-tag semplificati. Ad esempio si pensi solo all'abbandono dell'obbligatorietà dello slash finale nei tag standalone. In Figura 3.4 viene riportata una comparazione tra la prima riga di codice che viene inserita in un documento HTML4 ed in un documento HTML5.

Figura 3.4: *Due spezzoni di codice che fanno la stessa cosa, in HTML4 e HTML5 rispettivamente.*

Oltre alla semplificazione dell'utilizzo di meta-tag che erano presenti nelle versioni precedenti del linguaggio, HTML5 presenta anche delle novità: sono stati introdotti infatti dei tag prototipizzati dai più comuni tipi di pagine web e servizi. Per quanto riguarda la struttura della pagina sono stati definiti i tag <header>, <nav>, <main> e <footer>, visibili in Figura 3.5.

Inoltre, per i contenuti tipici di pagine web quali blog, stampa ed e-commerce sono stati definiti i meta-tag <article>, <section>, <summary>, <details> e <time>, essenziali in questi tipi di contenuti. Di seguito uno spezzone di codice che illustra come questi nuovi tag semplificano la strutturazione di una pagina web senza l'ausilio di tag annidati, soluzione spesso

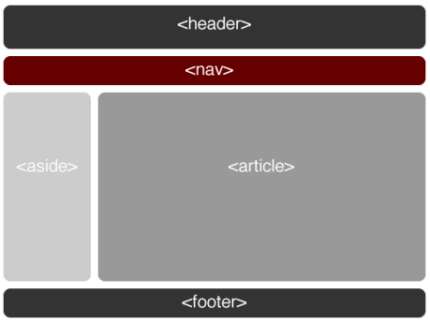

Figura 3.5: *Disposizione degli elementi in pagina.*

utilizzata dagli sviluppatori per definire una pagina web secondo le caratteristiche tipiche, e che procurava spesso problemi quando dovevano essere ricollocate le sezioni della pagina stessa.

```
<header>
 <hgroup>
  <h1>Page Title</h1>
  <h2>Tag Line text...</h2>
 </hgroup>
</header>

<nav><ul>...list of nav links...</ul></nav>

<aside>...sidebar links...</aside>

<article>
 <header>
  <h1>Article Title</h1>
  <time datetime="2012-06-09" pubdate">Published on June 9, 2012.</time>
 </header>
 <p>Article Content</p>
 <footer>...comments, permalink, etc...</footer>
```

Figura 3.6: *Codice rappresentante una pagina suddivisa in elementi.*

Inoltre, per venire incontro alla sempre maggiore diffusione di elementi multimediali presenti all'interno delle pagine web, cosa che avveniva con l'ausilio di plugin embedded e specifici (con l'onere da parte dell'utente finale di installazione nel proprio browser dei suddetti plugin), sono stati introdotti i meta-tag <audio> e <video>: di seguito uno spezzone del loro impego all'interno del codice HTML

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

Figura 3.7: *Codice per l'inserimento di elementi audio e video in pagina.*

3.3.1 HTML5 Custom Data Attributes (HTML5 data-*)

Una delle novità più importanti in HTML5 è stata l'introduzione degli attributi custom (tag <*data-**>): essendo stato progettato ai fini di una maggiore estensibilità per i dati che devono essere associati ad un particolare elemento, che però non presenta un significato ben definito, è divenuto naturale per gli sviluppatori poter definire ed associare uno specifico tag a tali elementi. Questi custom data attributes consentono di memorizzare informazioni aggiuntive sullo standard, elementi HTML semantici senza workaround quali attributi non standard, utilizzo di proprietà extra sul DOM. In Figura 3.8 un esempio di utilizzo di tali tag.

```
<ul>
    <li id="coffee-12" data-single_origin="true">Hyper roast</li>
</ul>
```

```
var isSingleOrigin = coffee12.dataset.single_origin;
```

```
var coffee12IsSingleOrigin = $("#coffee-12").data("single_origin");
```

Figura 3.8: *Utilizzo dei custom tag data-*.*

L'utilizzo dei tag custom data avviene in maniera molto semplice: lo sviluppatore definisce tali tag direttamente all'interno della pagina HTML, e li valorizza con un valore che può essere una stringa di qualsiasi tipo. In tale modo si ha quindi una coppia *Nome-Valore* associata al nuovo tag che memorizza informazioni aggiuntive nella pagina, informazioni che non sarebbe stato possibile salvare in tag predefiniti: tramite l'utilizzo del prefisso *data-* il browser ignora la presenza di questi tag, nascondendo di fatto tali informazioni al browser ed all'utente finale; sarà infatti compito della parte JavaScript utilizzare tali informazioni per migliorare la user experience. Ad esempio possono essere utilizzati per memorizzare l'altezza iniziale o l'opacità di un elemento che potrebbe essere necessario conoscere in successivi calcoli per una animazione o per memorizzare i parametri per un filmato Flash che è caricato tramite JavaScript o, ancora, per memorizzare dati relativi ad un elemento di un gioco: senza l'utilizzo dei custom data tag questo imporrebbe allo sviluppatore di implementare un sistema di memorizzazione lato server con query su DB e chiamate Ajax per il recupero di tali dati.

3.3.2 HTML5 input types

In HTML5 l'approccio ai tipi di input nel form è semantico, questo ha degli indiscussi vantaggi:

- Collega in maniera coerente i campi del form ai tipi di dato presenti nelle tabelle di un db
- Trasporta informazioni essenziali per il processo di validazione
- Istruisce l'interprete (il browser) verso un corretto output.

Vengono ora presentati i tipi di input già presenti nelle versioni precedenti del linguaggio, a seguito, i nuovi tipi definiti in HTML5.

Input Type Text: *< input type="text" >* definisce un campo di testo di input su un'unica linea;

Input Type Password: *< input type="password" >* definisce un campo password;

Input Type Submit: *<input type="submit">* definisce un pulsante per l'invio di dati ad un form-handler. Quest'ultimo è tipicamente una server page contente uno script per l'elaborazione dei dati in input, definito dall'attributo action della form;

Input Type Reset: *<input type="reset">* definisce un pulsante di reset che resetta tutti i campi della form al loro valore di default;

Input Type Radio: *<input type="radio">* definisce un radio button, che permette all'utente di selezionare SOLO UNA possibile scelta tra una lista definita;

Input Type Checkbox: *<input type="checkbox">* deninisce una checkbox, che permette all'utente di selezionare ZERO o PIU' opzioni tra una lista di opzioni possibili;

Input Type Button: *<input type="button">* definisce un pulsante;

3.3.3 Nuovi tipi di input definiti in HTML5

Input Type Number
<input type="number"> definisce un campo di input di tipo numerico; tale campo può contenere anche una serie di restrizioni sui valori inseriti, come mostrato nella tabella seguente (l'icona di HTML5 specifica che sono attributi introdotti specificatamente in questa versione del linguaggio).

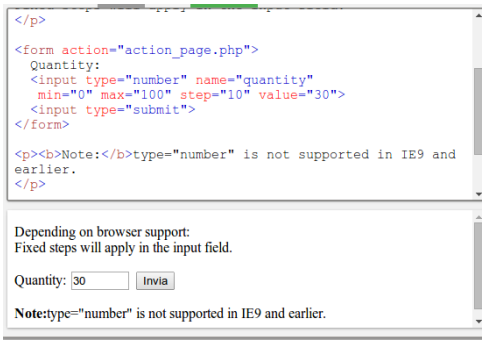

Figura 3.9: *Codice e rendering del tag <input type="number">.*

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

Figura 3.10: *Tabella delle restrizioni degli attributi di tipo numerico.*

Input Type Date

<input type="date"> viene usato per campi di input che contengono date; in base al browser utilizzato, può essere visibile un date picker nel campo.

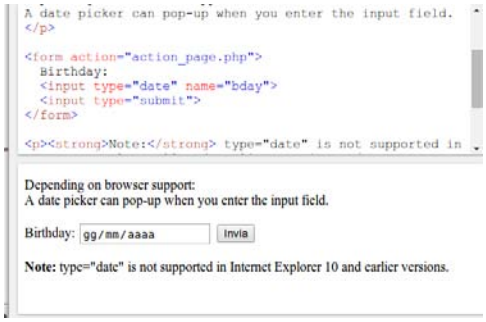

Figura 3.11: *Codice e rendering del tag <input type="date">.*

Input Type Color

<input type="color"> viene usato per i campi input che devono contenre un colore; in base al browser utilizzato, può essere visibile un color picker nel campo.

Figura 3.12: *Codice e rendering del tag <input type="color">.*

Input Type Range

<input type="range"> viene usato per campi di input che devono contenere un valore compreso in un determinato range; in base al browser utilizzato, il campo di input può essere visualizzato come uno slider.

Figura 3.13: *Codice e rendering del tag <input type="range">.*

Input Type Month

<input type="month"> permette all'utente di selezionare un mese ed un anno; in base al browser utilizzato, può essere visibile un date picker nel campo.

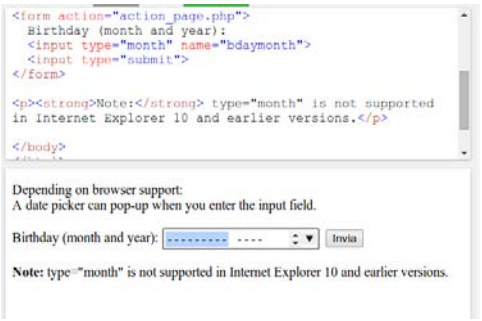

Figura 3.14: *Codice e rendering del tag <input type="month">.*

Input Type Week

<input type="week"> permette all'utente di selezionare una determinata settimana e un anno; in base al browser utilizzato, può essere visibile un date picker nel campo.

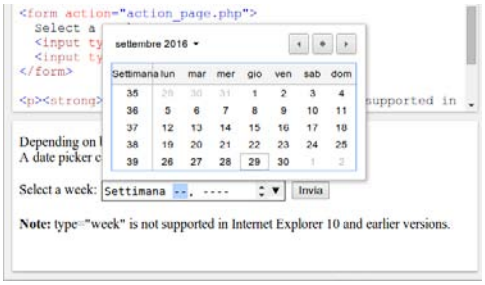

Figura 3.15: *Codice e rendering del tag <input type="week">.*

Input Type Time

<input type="time"> permette all'utente di selezionare un orario (non la time zone); in base al browser utilizzato, può essere visibile un time picker nel campo.

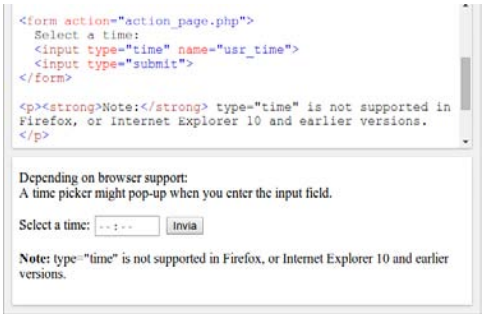

Figura 3.16: *Codice e rendering del tag <input type="time">.*

Input Type Datetime-local

<input type="datetime-local"> specifica un orario e una data nel campo di input (ma non la time zone); in base al browser utilizzato, può essere visibile un date picker nel campo.

Figura 3.17: *Codice e rendering del tag <input type="datetime-local">.*

Input Type Email

<input type="email"> viene usato per campi di input che devono contenere un indirizzo e-mail; in base al browser utilizzato, l'indirizzo e-mail inserito può venire validato automaticamente all'invio dei dati. Inoltre, alcuni smartphone, riconoscendo il tipo di campo, aggiungono i tasti "@" e ".com" alla tastiera virtuale.

Figura 3.18: *Codice e rendering del tag <input type="email">.*

Input Type Search

<input type="search"> viene usato per i campi di ricerca (anche se assomiglia ad un tipico campo di input).

Figura 3.19: *Codice e rendering del tag <input type="search">.*

Input Type Tel

<input type="tel"> viene usato per campi di input che contengono numeri telefonici; questo tipo di campo è attualmente supportato solo da Safari 8.

Figura 3.20: *Codice e rendering del tag <input type="tel">.*

Input Type Url

<input type="url"> viene usato per campi di input che contengono indirizzi URL;in base al browser utilizzato, l'indirizzo URL inserito può venire validato automaticamente all'invio dei dati. Inoltre, alcuni smartphone, riconoscendo il tipo di campo, aggiungono il tasto ".com" alla tastiera virtuale.

Figura 3.21: *Codice e rendering del tag <input type="url">.*

Capitolo 4

Bootstrap

4.1 Introduzione

Nel mondo del web, gli sviluppatori usualmente cominciano un progetto da un file index.html con qualche porzione di testo senza neanche un set del font da utilizzare.

Figura 4.1: *Il logo di Bootstrap.*

Del resto, qualche anno fa questo era tipico negli sviluppi di progetti che partivano da zero: fortunatamente i tempi sono cambiati. Da qualche anno si stanno sempre più affermando tra gli sviluppatori i cosiddetti "framework CSS". Si tratta di pacchetti di file (html, css, e anche javascript) che consentono di iniziare a sviluppare un frontend partendo già da una base solida, collaudata e standard.

Molti sviluppatori negli anni si sono preparati un pacchetto di default da utilizzare in fase di partenza di un nuovo progetto, salvandosi porzioni di codice o intere pagine utili ed evitando di dover partire da zero ogni volta: proprio sulla scia di questo "metodo lavorativo" anche il team di sviluppatori del colosso Twitter ha deciso di crearsi un proprio framework per allineare e standardizzare i vari progetti interni, e la cosa interessante è che nel 2011 hanno deciso di condividerlo con il resto del mondo, rilasciandolo come progetto Open Source chiamato appunto Bootstrap Twitter:

Bootstrap è un insieme di elementi grafici, stilistici, di impaginazione e Javascript pronti all'uso, nati internamente a Twitter ad opera degli sviluppatori Mark Otto e Jacob Thornton. Oggi Bootstrap è un progetto indipendente ed è stato messo a disposizione degli sviluppatori di tutto il mondo che sono liberi di utilizzare questo framework come base per i propri progetti web: il suo largo utilizzo nel mondo dello sviluppo di applicazioni responsive ne afferma quindi il titolo di uno dei principali framework usati; forte di una platea

di utilizzatori in rapida espansione, ad oggi Bootstrap rappresenta una delle soluzioni più utilizzate per la progettazione di template per il web, soprattutto in ottica responsiva.

Infatti Twitter Bootstrap è attualmente il framework front-end più diffuso che permette di creare rapidamente e facilmente siti web responsive con varie applicazioni, in quanto semplice, intuitivo e veloce. Bootstrap utilizza HTML, CSS e Javascript: essenzialmente è un insieme di strumenti gratuiti e pronti all'uso che consentono di utilizzare i suddetti tools con molta facilità e che si adattano al dispositivo che si sta utilizzando. Essenzialmente si compone di quattro gruppi di elementi:

- **CSS base**
- **Scaffolding**
- **Componenti**
- **JavaScript**.

CSS base: comprende tutti gli stili predefiniti dei diversi elementi come ad esempio i titoli, i pulsanti, le immagini, i form e quant'altro. Per utilizzare questi elementi è sufficiente applicare le diverse classi disponibili.
Scaffolding: consente di definire il layout della pagina mediante una griglia fissa o fluida con una larghezza base di 960px, denominata Grid System, mediante la quale è possibile allineare i contenuti. La griglia è costituita da 12 colonne, in cui ogni elemento può espandersi liberamente su più colonne. Un elemento è definito mediante una classe, ad esempio class=“col-md-1”. Per estenderla su due colonne è sufficiente modificare la classe in class=“col-md-2”.
Componenti: comprende le definizioni di pulsanti, tabelle, barre di navigazione, menu a discesa, ecc. Inoltre comprende una serie di caratteri di uso comune (ad esempio il pulsante “+”), resi disponibili da *glyphicons.com*. Anche questi caratteri/icone si utilizzano mediante classi:

```
< button type =“button” class =“btn btn-default btn-lg” >
< span class =“glyphicon glyphicon-volume-up”></ span >
</ button >
```

JavaScript: comprende diversi plug-in jQuery per realizzare effetti come finestre modali, popup, transizioni, ecc. La cosa interessante di Bootstrap è il fatto che non è necessaria una conoscenza approfondita nè di HTML, nè dei fogli di stile CSS: per poter utilizzare questo sistema basterà conoscere solo le basi di tali linguaggi. Questo però non ne giustifica l'utilizzo in qualsiasi contesto: lo scopo di Bootstrap è infatti dare un punto di partenza (come implicitamente suggerito dal nome stesso del framework) per lo

sviluppo di progetti web dal design convenzionale ma con solide basi nella struttura dello stesso. Si può quindi considerare Bootstrap un toolkit (più che un framework) per gestire al meglio la fase di avvio di un progetto, un modo per poter contare su una serie di componenti riusabili e personalizzabili, comunque adattabili in termini stilistici ed estetici alle richieste del progetto e alla creatività di chi concepisce il sito. Secondo alcuni questa struttura presenta delle limitazioni, ma che può avere una sua funzione e una sua utilità per non dover ricominciare ogni volta da zero: infatti questo toolkit/framework offre i mattoni con cui costruire pagine web HTML5, completamente responsive, coerenti e funzionali. L'utilità di Bootstrap è immediatamente evidente, soprattutto nella situazione attuale in cui le pagine web possono essere fruite su un'ampia gamma di dispositivi con caratteristiche diverse. Sarà Bootstrap ad occuparsi di mettere a disposizione elementi di stile che permettono alla pagina di adattarsi al dispositivo utilizzando, al contempo, elementi di interfaccia comuni ai siti moderni, quelli cioé che l'utente si aspetta e di cui conosce comportamento e significato. Bootstrap, come ogni altro framework, consente di sviluppare un progetto in modo più rapido, tuttavia non può e non deve essere considerato la soluzione definitiva ad ogni problema di web-design. Così come non è sempre corretto fare ricorso ad un plugin di Javascript come jQuery, lo stesso vale per Bootstrap: l'opportunità o meno di un suo utilizzo deve essere valutata caso per caso in base alle specifiche esigenze di sviluppo.

4.2 Storia

Nel 2011, Bootstrap è stato creato come soluzione interna per ridurre le incoerenze nello sviluppo tra i membri del team di ingegneri di Twitter. Fondamentalmente, non esisteva una struttura di codice precisa che gli ingegneri erano tenuti a seguire per lo sviluppo della piattaforma. Lo sviluppo e la progettazione web sono un'arte, nonostante il disaccordo di molti, e ogni ingegnere ha il proprio modo di fare le cose. Questo può andare bene in alcuni casi, ma quando più ingegneri lavorano allo stesso progetto con approcci leggermente diversi, le incoerenze sono inevitabili. Le incoerenze nella progettazione web possono portare a problemi di codifica fondamentali che creano incertezza e aumentano il tempo necessario per la manutenzione. Bootstrap è uno strumento sviluppato originariamente da Mark Otto e Jacob Thorton, rispettivamente un designer e un ingegnere di Twitter, ed aveva come obiettivo quello di spingere il team di ingegneri di Twitter a utilizzare lo stesso framework per minimizzare le incoerenze. Va da sé che l'iniziativa Bootstrap riuscì a velocizzare il processo di lavoro e l'efficienza dell'intero team con meno incoerenze. Nonostante sia nato come una soluzione interna a Twitter, Mark e Jacob si resero presto conto che avevano spianato la strada a qualcosa di molto più significativo. Nell'agosto del 2011, il framework

Bootstrap è stato lanciato come progetto open-source su Github. Nel giro di pochi mesi, migliaia di sviluppatori di tutto il mondo contribuirono al codice e Bootstrap diventò il progetto di sviluppo open-source più attivo del mondo. Da allora, Bootstrap ha acquisito notorietà ed è diventato "il framework front-end più popolare per lo sviluppo di progetti responsivi, mobile-first sul web".

Figura 4.2: *Bootstrap si adatta ad ogni dispositivo.*

Dalla versione 2 in poi, Bootstrap è inoltre diventato il framework simbolo del responsive design, grazie alla sua impostazione particolarmente focalizzata sull'utilizzo di griglie mobile friendly.

4.3 Specifiche

Come precedentemente detto, possiamo suddividere i suoi elementi in quattro macro-aree:

4.3.1 Scaffolding (o impalcatura)

Questa area contiene tutti quegli elementi CSS che permettono di definire la struttura della pagina, ossia il suo layout. La parte costitutiva di questa pagina è il Grid System, ossia una griglia, fissa o fluida, con una larghezza base di 960px nella quale possono essere definite righe e colonne in cui poter incasellare i contenuti. Tale sistema, molto caro ai grafici, permette di avere un sistema di riferimento per creare dei design proporzionati, efficaci e armoniosi: in questo modo quando è necessario decidere la dimensione di un certo elemento, si può ragionare non più in termini assoluti (pixel, punti o altro) ma in termini appunto di "colonne". Questo consente di avere un naturale allineamento delle diverse sezioni e di dividere lo spazio senza dover calcolare ogni volta le dimensioni da assegnare. Bootstrap utilizza un Grid

System a 12 colonne da 60 px ognuna e distanziate di 20 px tra loro, inoltre ha anche delle classi per gestire i Layout Fluidi, ovvero con dimensioni in percentuale anziché fisse. Un elemento per cui è definita la classe *class="col-md-2"*, ad esempio, occuperà automaticamente 2 colonne della griglia base. Perché si estenda su quattro colonne basterà modificare la classe precedente in *class="col-md-4"*.

Figura 4.3: *Suddivisione in colonne della pagina secondo il Grid System.*

4.3.2 CSS base

Bootstrap definisce un gran numero di classi CSS, per dare un aspetto gradevole a molti elementi usati spesso se non sempre: infatti questa parte contiene degli stili predefiniti per diversi elementi della pagina, come i titoli (H1, H2,...), le tabelle, i pulsanti, gli elementi dei form, le immagini... Con queste regole di stile realizzare pulsanti di varie dimensioni, con i bordi smussati, con un effetto over o altro diventa davvero semplice e immediato. Lo stesso vale per la creazione di una tabella con righe a colori alterni. Nel primo caso, basta applicare una delle moltissime classi disponibili, come *class="btn btn-primary"* o *class="btn btn-default btn-lg"*, nel secondo, basta applicare alla tabella la classe *class="table table-striped"*. Un altro esempio è l'applicazione di uno stile aggiungendo semplicemente *class="btn"* ad un anchor link o ad un input button, rendendolo molto più piacevole dal punto di vista grafico. Anche per le tabelle, i form, i titoli, le immagini vale la stessa cosa. Non è da sottovalutare inoltre, che usando Bootstrap si è abbastanza sicuri che non ci saranno problemi di compatibilità cross-browser, perché il framework è abbondantemente testato e ottimizzato.

4.3.3 Componenti

Oltre che dal punto di vista grafico, Bootstrap aiuta molto lo sviluppatore anche nell'implementazione di controlli e sviluppo di elementi dinamici delle pagine. Si consideri ad esempio menu dropdown, interfacce a tab, tooltip, alert, menu ad accordion, slider, banner di navigazione e molto altro: infatti questa area contiene elementi più complessi di pulsanti o tabelle, ma ormai molto comuni nei siti web. Fra i componenti è compreso anche un set di icone, o meglio di glifi (dato che non si tratta di immagini ma di caratteri) di uso comune, messe a disposizione da *glyphicons.com*. Le icone/glifi Glyphicons di norma sono a pagamento, mentre il test distribuito con Bootstrap è gratuito. I creatori di Bootrstap suggeriscono a chi le usa di inserire, se possibile, un link al sito Glyphicon, come una sorta di ringraziamento. Tali icone si utilizzano tramite apposite classi; ad esempio per creare un pulsante base con l'icona di una stella è sufficiente creare il pulsante con la sua classe e al suo interno inserire l'icona che occorre con elemento di una apposita classe:

```
<button type="button" class="btn btn-default btn-lg">
<span class="glyphicon glyphicon-star"></span>
</button>
```

Questo meccanismo delle classi si adotta anche per gli elementi più complessi. Ad esempio per creare un menu a discesa basta usare una elenco non ordinato con apposite voci:

```
<div class="dropdown">
<ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1">
<li role="presentation"><a role="menuitem" tabindex="-1" href="">Uno</a></li>
<li role="presentation"><a role="menuitem" tabindex="-1" href="">Due</a></li>
<li role="presentation"><a role="menuitem" tabindex="-1" href="">Tre</a></li>
<li role="presentation" class="divider"></li>
<li role="presentation"><a role="menuitem" tabindex="-1"href="">A</a></li>
</ul>
</div>
```

4.3.4 Javascript

Quest'ultima area contiene diversi plug-in jQuery per realizzare effetti molto comuni come transizioni, finestre modali, popup, carousel, accordion, tab. Anche questi plug-in sono semplici da usare e permettono di realizzare tantissime soluzioni interessanti. Spesso non è necessario neanche scrivere una sola riga di codice Javascript per ottenere i risultati desiderati, infatti basta usare i cosidetti *data-attributes*, cioé degli attributi particolari da aggiungere ai tag html, che lo script di base di Bootstrap interpreta e gestisce senza nessun intervento da parte dello sviluppatore. Ad esempio per creare un tooltip su un link, basterà aggiungere gli attributi *rel="tooltip" title="first tooltip"* nel TAG <a>.

4.3.5 Esempi di pagine bootstrap su dispositivi differenti

Nelle prossime figure si può notare il rendering di una pagina web sviluppata con Bootstrap su differenti dispositivi: su pc desktop, su tablet e su smartphone, rispettivamente.

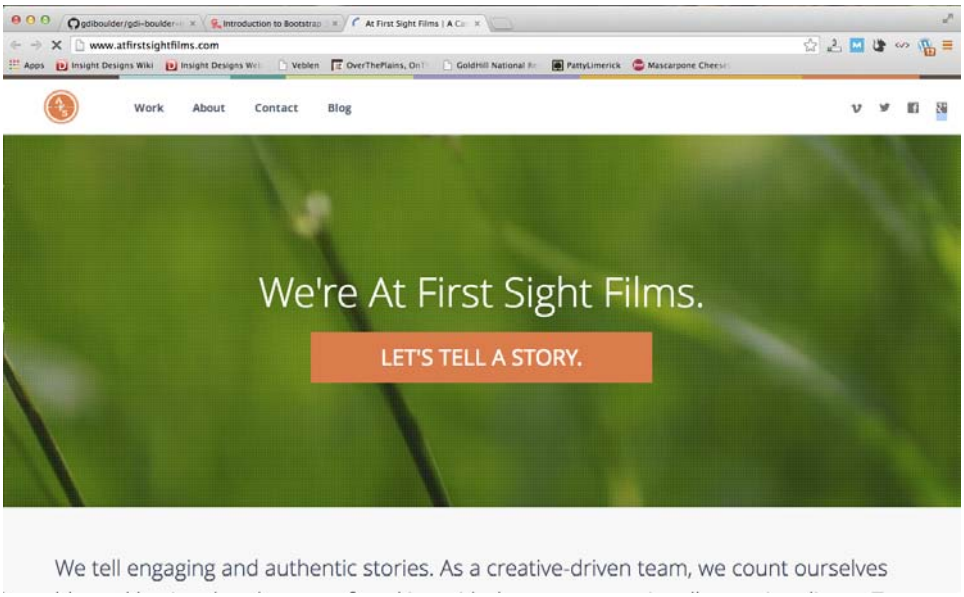

Figura 4.4: *Il rendering del sito www.atfirstsightfilms.com su desktop pc.*

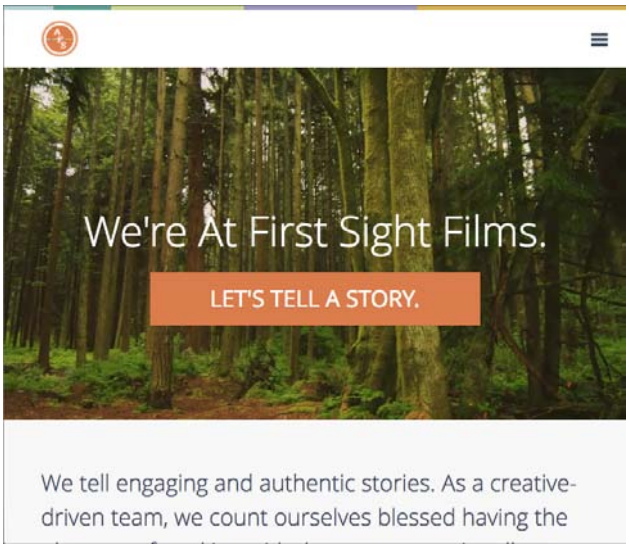

Figura 4.5: *Il rendering del sito www.atfirstsightfilms.com su tablet.*

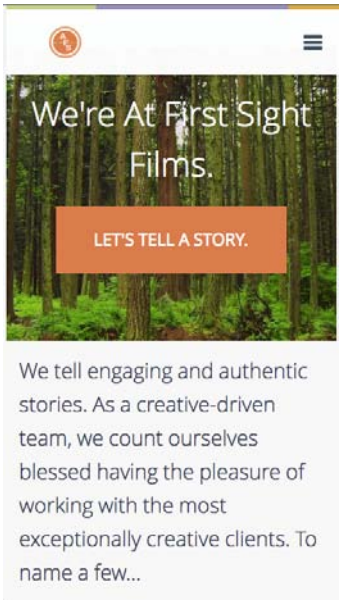

Figura 4.6: *Il rendering del sito www.atfirstsightfilms.com su smartphone.*

4.4 Versione 3 e novità

4.4.1 Responsive design e griglia

L'ultima versione stabile di Bootstrap è la 3.3.7, ed è stata rilasciata il 25 luglio 2016: in questo paragrafo verranno illustrate le differenze che la versione 3 ha introdotto rispetto al rilascio precedente. La principale novità nell'ultima versione di Bootstrap è senza dubbio rappresentata dal fatto che ora il framework è concepito nativamente come uno strumento adatto alla realizzazione di siti responsivi. Si può così sintetizzare: se prima le funzionalità responsive erano opzionali, ora si è abbracciato l'approccio mobile first. Una pagina basata su Bootstrap è di default adattabile ad un'ampia gamma di device, dagli smartphone fino ai desktop ad altissima risoluzione. In questo scenario, il componente che ha subìto le modifiche più radicali è la griglia. In Bootstrap 3.0 il layout è di default fluido, con larghezze espresse in percentuale, e pertanto maggiormente adattabile a differenti risoluzioni di schermo. Una serie di classi speciali abbinate a specifici breakpoint e di funzionalità avanzate rendono possibile la realizzazione di layout complessi e robusti in ogni scenario dettato dal dispositivo in uso. Nel caso si volesse disabilitare le funzionalità responsive e operare con una griglia fissa sono disponibili sul sito ufficiale le istruzioni per implementare tale soluzione.

4.4.2 Altre novità

Le altre novità introdotte sono più di dettaglio. Vengono riportate sinteticamente, non prima di ricordare che nella documentazione ufficiale del framework è presente una sezione utile a chi effettua la migrazione dalle versioni 2.x del framework alla 3.0.

- Le icone che è possibile associare al design dell'interfaccia sono sempre quelle di Glyphicons, ma invece che essere rappresentate da immagini sono ora distribuite sotto forma di icon font
- È stato completamente riscritto il componente Navbar
- Sono stati introdotti nuovi componenti estremamente flessibili come i pannelli e i list groups. Altri componenti sono stati eliminati, altri ancora ridefiniti nell'aspetto visuale
- Adeguandosi in parte alla tendenza del flat design, Bootstrap 3.0 si presenta nel suo look&feel predefinito con una veste estetica più semplificata
- Sul versante Javascript, è praticamente invariato il set di componenti disponibili. I plugin sono però stati riscritti per ottimizzare ulteriormente le prestazioni, per evitare conflitti tra script, per adeguare alcuni componenti al nuovo contesto responsivo

- A livello di supporto per l'utente, sono da evidenziare una documentazione più ampia e un pannello di personalizzazione notevolmente più ricco nelle opzioni offerte.

4.4.3 Supporto dei browser

Per quanto concerne il supporto dei vari browser va evidenziato che con questa release Bootstrap non presenta supporto per Internet Explorer 7 e Firefox 3.6 (e versioni precedenti). Ufficialmente il framework supporta le ultime versioni di Chrome, Safari, Firefox, Opera e Internet Explorer. Per Internet Explorer 8 il supporto pieno è garantito solo adoperando lo script Respond.js in grado di aggiungere a tale browser la capacità di gestire le media query. Altre funzionalità minori del framework non sono supportate su IE9.

4.5 Un esempio pratico

Essendo lo scopo dell'attuale tesi quello di illustrare il funzionamento di Bootstrap nello sviluppo di pagine web responsive si mostra in questo paragrafo un semplice esempio di progettazione.

Quello che si vuole ottenere è una homepage con le seguenti caratteristiche:

- un layout responsivo
- un menu responsivo usando il modulo integrato
- un carousel completo.

Di seguito un'anteprima del layout finito

Figura 4.7: *Il layout del sito.*

Per prima cosa viene creata una pagina alla quale collegare i file necessari per poter iniziare a costruire il layout. Oltre a Bootstrap vengono utilizzati jQuery, html5shiv e respond, quindi si procede al download:

- **Bootstrap 3** può essere scaricato in diversi modi, ad esempio utilizzando Bower o NPM. In questo esempio viene scaricato l'archivio con i file già compilati
- **jQuery** è necessario per poter utilizzare la parte javascript del framework jQuery 1.11.2 e relativo file .map (per editare i CSS generati)
- **html5shiv** è un piccolo script che garantisce la retro compatibilità dei tag HTML5 con IE8 e precedenti
- **Respond** garantisce il funzionamento delle media query anche su IE8 e precedenti.

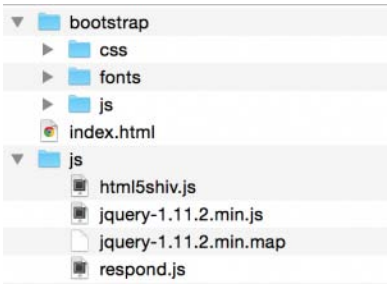

Figura 4.8: *La struttura iniziale del progetto.*

Viene creata una nuova pagina web (denominata index.html) in HTML5 così strutturata:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=
<title>Il mio primo layout in bootstrap</title>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootst
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.js"></script>
<![endif]-->
</head>

<body>

<script src="js/jquery-1.11.2.min.js"></script>
<script src="bootstrap/js/bootstrap.min.js"></script>
</body>
</html>
```

Figura 4.9: *La pagina index.html.*

Questa pagina può essere un ottimo punto di partenza per tutti i progetti con Bootstrap perché garantisce un funzionamento ottimale del framework e una buona retro compatibilità con i vecchi browser di casa Microsoft. Una volta definita la pagina di partenza viene strutturato il layout definendo le aree principali come l'header, l'area dedicata al carousel o altro. Qui di seguito il contenuto del body:

```
<div id="headerWrap" class="wrap">
    <header class="container">
        <nav id="mainNav">
        <!-- qui il menù principale -->
        </nav>
    </header>
</div>
<div class="container">
    <div id="carousel"></div>
    <div id="content"></div>
</div>
<div id="footerWrap" class="wrap">
    <footer class="container">
    <nav id="tools">
    <!-- qui il menù secondario -->
    </nav>
    </footer>
</div>
```

Figura 4.10: *Il contenuto del body.*

Il layout è suddiviso in quattro parti: l'header, il carousel, la sezione di contenuto che conterrà tre widget organizzati tramite la griglia di Bootstrap (illustrata più avanti) e il footer.

4.5.1 Struttura dell'header

Per prima cosa viene illustrato lo sviluppo dell'header all'interno del quale viene inserito il primo componente: il menu responsivo. Questo componente si adatta ottimamente allo spazio disponibile secondo i quattro breakpoint che Bootstrap mette a disposizione (large, medium, small e extra-small) con le opportune media query. Assieme al componente viene inserito il logo testuale a sinistra, mentre a destra viene posizionato il campo di ricerca. Di seguito il contenuto dell'header:

```
<nav id="mainNav" class="navbar navbar-inverse" role="navigation">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" s> <span class="sr-only">Toggle
navigation</span> <span class="icon-bar"></span> <span class="icon-bar"></span> <span class="icon-bar"></span> </button>
        <a href="#" class="navbar-brand">WOW</a> </div>
      <div class="collapse navbar-collapse" id="mainMenu">
        <ul class="nav navbar-nav">
          <li><a href="#">Chi siamo</a></li>
          <li class="dropdown">
            <a class="dropdown-toggle" data-toggle="dropdown" role="button" aria-expanded="false" href="#">Servizi <span class="care
t"></span></a>
            <ul class="dropdown-menu" role="menu">
              <li><a href="#">Ideazione loghi</a></li>
              <li><a href="#">Immagine coordinata</a></li>
              <li class="divider"></li>
              <li><a href="#">Sviluppo siti web</a></li>
              <li><a href="#">Creazione app.</a></li>
            </ul>
          </li>
          <li><a href="#">Portfolio</a></li>
          <li><a href="#">Contatti</a></li>
        </ul>
        <form class="navbar-form navbar-right" role="search">
          <div class="form-group">
            <input type="text" class="form-control" placeholder="Cerca nel sito...">
          </div>
        </form>
      </div>
</nav>
```

Figura 4.11: *Il contenuto dell'header.*

Il tag <nav> delimita il menu (navbar e navbar-inverse conferiscono forma e colore) dopodiché possono essere individuate due aree distinte al suo interno: il div con classe *"navbar-header"* e il div con id *"mainMenu"*. Il primo contiene logo e pulsante per l'attivazione del menu verticale in modalità extra small, mentre il secondo contiene una lista non ordinata e il campo di ricerca (ovvero gli elementi che dovranno essere inseriti nella tendina a scomparsa):

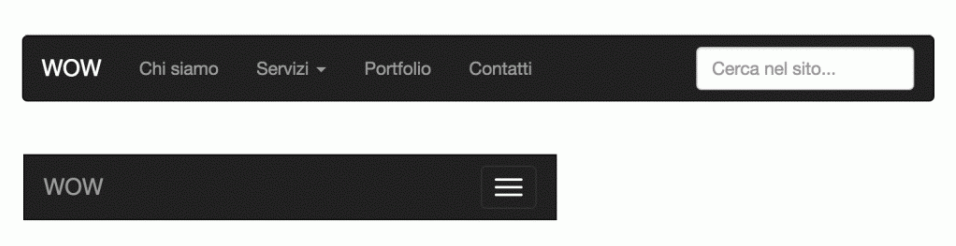

Figura 4.12: *Il rendering del menu.*

Il componente menu utilizza *collapse.js* , un plugin di Bootstrap incluso in *bootstrap.min.js* ; il button *"toggle navigation"* è collegato al menu principale e lo fa funzionare grazie alla sintassi dichiarativa *data-target="mainMenu"*

con la quale viene indicato la seconda sezione del menu. A questo punto però sorge un problema: il menu ha troppe voci per essere visualizzato su una sola riga nelle viste medium e small. La soluzione più elegante è modificare il comportamento del modulo e fare in modo che quest'ultimo collassi già in modalità small o addirittura medium. Si sconsiglia di modificare direttamente il framework, l'approccio migliore è ricompilare le sorgenti LESS modificando *@navbarCollapseWidth*. Una soluzione alternativa è scrivere una media query che sovrascriva il comportamento di default; quello riportato ad esempio attiva il menu a tendina in modalità small:

```
@media (max-width: 990px) {
    .navbar-header {
        float: none;
    }
    .navbar-toggle {
        display: block;
    }
    .navbar-collapse {
        border-top: 1px solid transparent;
        box-shadow: inset 0 1px 0 rgba(255,255,255,0.1);
    }
    .navbar-collapse.collapse {
        display: none!important;
    }
    .navbar-nav {
        float: none!important;
        margin: 7.5px -15px;
    }
    .navbar-nav>li {
        float: none;
    }
    .navbar-nav>li>a {
        padding-top: 10px;
        padding-bottom: 10px;
    }
}
```

Figura 4.13: *Il codice per il rendering alternativo tramite media query.*

Una seconda soluzione può essere nascondere alcune voci di menu in base alla risoluzione grazie alle responsive utilities che Bootstrap mette a disposizione:

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
`.visible-xs-*`	Visible	Hidden	Hidden	Hidden
`.visible-sm-*`	Hidden	Visible	Hidden	Hidden
`.visible-md-*`	Hidden	Hidden	Visible	Hidden
`.visible-lg-*`	Hidden	Hidden	Hidden	Visible
`.hidden-xs`	Hidden	Visible	Visible	Visible
`.hidden-sm`	Visible	Hidden	Visible	Visible
`.hidden-md`	Visible	Visible	Hidden	Visible
`.hidden-lg`	Visible	Visible	Visible	Hidden

Figura 4.14: *Le responsive utilities e il relativo comportamento su device differenti.*

4.5.2 Inserimento carousel

Il secondo componente che viene inserito è il carousel responsivo proposto fra gli strumenti predefiniti di Bootstrap. Il componente è semplice quanto duttile. Qui viene illustrato come inserire un carousel minimale collegato a tre placeholder. Di seguito carousel e il suo contenuto:

```
<div id="carousel"  class="carousel slide" data-ride="carousel">
    <!-- indicatori -->
    <ol class="carousel-indicators">
      <li data-target="#carousel" data-slide="0" class="active"></li>
      <li data-target="#carousel" data-slide="1"></li>
      <li data-target="#carousel" data-slide="2"></li>
    </ol>
    <!--qui le slide-->
    <div class="carousel-inner">
      <div class="item active"><img src="http://placehold.it/1140x500/ff0000/000000"/> </div>
      <div class="item"><img src="http://placehold.it/1140x500/0000ff/000000"/> </div>
      <div class="item"><img src="http://placehold.it/1140x500/00ff00/000000"/> </div>
    </div>
    <!-- i controlli -->
    <a class="left carousel-control" href="#carousel" role="button" data-slide="prev"> <span class="glyphicon glyphicon-chevron-
left"></span></a>
    <a class="right carousel-control" href="#carousel" role="button" data-slide="next"> <span class="glyphicon glyphicon-chevron-righ
t"></span></a>
    </div>
```

Figura 4.15: *Il componente carousel.*

Il carousel è strutturato in tre parti: gli indicatori, le slide (l'item può contenere anche una didascalia) e i controlli (se omessi si avrà uno slider

automatico). Usando la sintassi dichiarativa si può configurare il carousel semplicemente seguendo il seguente schema. Brevemente:

- la classe “carousel” applica il componente
- la classe “slide” conferisce l’animazione di scorrimento orizzontale (senza si passerebbe da una slide all’altra senza animazioni)
- data-ride=“carousel” provvede all’animazione automatica di passaggio da una slide all’altra, in sua assenza si devono utilizzare i pulsanti.

E’ possibile avere un controllo più specifico del comportamento del carousel con i seguenti attributi “data-”:

- data-interval: indica l’ammontare di tempo che intercorre fra una slide e l’altra (di default data-interval=5000)
- data-pause: attiva o meno la pausa del carousel quando il puntatore passa sulle slide (data-pause=“hover”)
- data-wrap: indica se il loop delle foto è o meno continuo o c’è una pausa al completamento delle slide (data-pause:true)
- data-keyboard: attiva o meno il controllo del carousel da tastiera.

4.5.3 Il content e la griglia

Sotto al carousel viene inserita una griglia Bootstrap su due righe per gestire una citazione e tre widget. La griglia è organizzata in righe (.row) e colonne (col-x-y). La griglia di Bootstrap è di 12 colonne per 4 viste per cui esistono 12 classi per ciascuna vista. In questo modo grazie alle media query è possibile gestire l’ampiezza di ciascuna colonna in base alla dimensione della finestra. Quindi se in modalità large (lg) si vuole ottenere una struttura a due colonne si deve scrivere:

```
<div class="row">
    <div class"col-lg-6"></div>
    <div class"col-lg-6"></div>
</div>
```

Figura 4.16: *Suddivisione su due colonne in modalità large (lg).*

In questo caso quando viene passato il breakpoint relativo alla vista medium, dato che non vengono utilizzate classi per le colonne della griglia corrispondente, le due colonne verranno settate al 100% e al posto di due colonne si avranno due righe.

Per gestire dimensioni differenti per la vista medium (ad esempio una dimensione di 8/12 per la colonna sinistra e 4/12 per la destra vengono aggiunte le opportune colonne come mostrato in Figura 4.17.

```
<div class="row">
    <div class"col-lg-6 col-md-8"></div>
    <div class"col-lg-6 col-md-4"></div>
</div>
```

Figura 4.17: *Suddivisione in due colonne in modalità medium (md) con diverse proporzioni.*

Se invece si vuole mantenere due colonne uguali sia in small, medium che large è sufficiente inserire le classi che regolamentano le colonna nella vista a risoluzione più bassa come mostrato in Figura 4.18.

```
<div class="row">
    <div class"col-sm-6"></div>
    <div class"col-sm-6"></div>
</div>
```

Figura 4.18: *Suddivisione delle colonne in modalità small (sm), e di conseguenza in ogni modalità.*

In questo caso si ottengono due righe solamente nella vista extra small. Ora viene illustrato il funzionamento relativo alla parte del contenuto (content). Al suo interno vengono inserite due righe così strutturate:

```
<div class="row voffset4">
      <div id="citation" class="col-md-6 text-center col-md-offset-3">
        <blockquote>“Il design non è come sembra o come appare. Il design è come funziona”.
          <footer><cite>Steve Jobs</cite></footer>
        </blockquote>
      </div>
    </div>
<div class="row voffset4" id="widgets">
      <div class="col-md-4 col-xs-6">
        <h3 class="text-center">Title</h3>
        <img class="img-thumbnail img-responsive" src="http://placehold.it/360x200/cccccc/000000"/>
        <p class="voffset2">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce justo metus, accumsan viverra iaculis in, pu
        <a class="btn btn-default pull-right voffset2">Leggi tutto...</a> </div>
      <div class="col-md-4 col-xs-6">
        <h3 class="text-center">Title</h3>
        <img class="img-thumbnail img-responsive" src="http://placehold.it/360x200/cccccc/000000"/>
        <p class="voffset2">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce justo metus, accumsan viverra iaculis in, pu
        <a class="btn btn-default pull-right voffset2">Leggi tutto...</a> </div>
      <div class="col-md-4 hidden-sm hidden-xs">
        <h3 class="text-center">Title</h3>
        <img class="img-thumbnail img-responsive" src="http://placehold.it/360x200/cccccc/000000"/>
        <p class="voffset2">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce justo metus, accumsan viverra iaculis in, pu
        <a class="btn btn-default pull-right voffset2">Leggi tutto...</a> </div>
    </div>
</div>
```

Figura 4.19: *Codice relativo al contenuto del sito.*

La prima riga contiene una sola cella a cui è attribuito un col-md-6 e un col-md-offset-3: questo significa che la colonna da 6 verrà centrata e il suo contenuto occuperà la metà dello spazio disponibile (in questo modo la citazione sarà disposta su 2 righe in tutte le viste a parte la xs. Il comportamento sarà identico per la vista medium e large mentre nella vista small e extrasmall la cella andrà a 100% in assenza di regole definite. La seconda riga contiene 3 widget costituiti da titolo (h3), immagine e testo. Qui il comportamento prevede 3 colonne per la vista large e media (ottenuta utilizzando col-md-4) mentre per la vista small e extra small viene nascosta l'ultima colonna con hidden-xs e hidden-sm e settando 2 colonne usando col-xs-6 sulle celle rimanenti. In questa parte di codice sono state utilizzate inoltre alcune classi di supporto e alcune custom non facenti parte del framework (.voffsetX).I titoli h3 sono stati centrati con la classe di supporto text-center. Le immagini sono state rese responsive usando img-responsive mentre con img-thumbnail sono stati aggiunti i bordi bianchi arrotondati. Ogni widget ha un link di approfondimento, le classi che lo formattano sono:

- btn e btn-default che gli conferiscono l'aspetto di un pulsante
- pull-right che lo flottano a destra.

4.5.4 Inserimento footer

La pagina viene conclusa inserendo un menù nel footer. In questo caso non sarà necessario usare la griglia:

```
<div id="footerWrap" class="voffset5">
  <footer class="container">
    <p class="text-center">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusc
  </footer>
</div>
```

Figura 4.20: *Suddivisione delle colonne in modalità small (sm), e di conseguenza in ogni modalità.*

4.5.5 Il foglio di stile personalizzato

Benché si sia cercato di utilizzare esclusivamente quanto messo a disposizione dal framework sono stati necessari alcuni stili che sono stati inseriti in un file denominato style.css (inclusi subito dopo a quello di Bootstrap) mostrato in Figura 4.21.

```
/* modifica allo stile di default per la citazione */
#citation blockquote{ font-size:2.4rem; border-left: none;}

/* offset verticali */
.voffset  { margin-top: 2px; }
.voffset1 { margin-top: 5px; }
.voffset2 { margin-top: 10px; }
.voffset3 { margin-top: 15px; }
.voffset4 { margin-top: 30px; }
.voffset5 { margin-top: 40px; }
.voffset6 { margin-top: 60px; }
.voffset7 { margin-top: 80px; }
.voffset8 { margin-top: 100px; }
.voffset9 { margin-top: 150px; }

/* stile per il footer e il suo wrapper */
#footerWrap { background-color:#222222; padding:40px 0;}
#footerWrap p { color:#fff;}
```

Figura 4.21: *La classe di stile con alcune personalizzazioni.*

Questo paragrafo aveva come obiettivo fornire una panoramica generale dell'utilizzo di bootstrap attraverso la creazione di una homepage: è stato possibile realizzare un sito web responsive in maniera piuttosto semplice, essendo Bootrap praticamente una guida per lo sviluppo, ed occupandosi in maniera piuttosto limitata del funzionamento responsivo, lasciando allo sviluppatore il solo compito di decidere quale contenuto mettere in rete. Inoltre Boostrap ha molti layout già pronti e scaricabili gratuitamente, che possono essere usati da chiunque voglia creare un sito web secondo gli ultimi standard di responsive design anche senza essere un perfetto conoscitore dei linguaggi di sviluppo.

Capitolo 5

AngularJS

5.1 Introduzione

AngularJS è un framework strutturale per applicazioni web dinamiche JavaScript, patrocinato da Google, utile a semplificare la realizzazione di applicazioni Web single page tramite l'utilizzo dei tag HTML come linguaggio di template che permette di estenderne la sintassi per esprimere i componenti della applicazione in modo chiaro e conciso. Con Single-page application (o in sigla SPA) si intende una applicazione web o un sito web che può essere usato o consultato su una singola pagina web con l'obiettivo di fornire un'esperienza utente più fluida e simile alle applicazioni desktop dei sistemi operativi tradizionali.

In un'applicazione su singola pagina tutto il codice necessario (HTML, JavaScript e CSS) è recuperato in un singolo caricamento della pagina o le risorse appropriate sono caricate dinamicamente e aggiunte alla pagina quando necessario, di solito in risposta ad azioni dell'utente. La pagina non si ricarica in nessun punto del processo, né lascia il controllo in un'altra pagina, sebbene moderne tecnologie web di HTML5 possano fornire la percezione e la navigabilità in pagine logiche separate nell'applicazione.

L'interazione con un'applicazione a singola pagina spesso coinvolge una comunicazione dinamica con il web server: ciò favorisce un approccio dichiarativo allo sviluppo client-side, migliore per la creazione di interfacce utente, laddove l'approccio imperativo è ideale per realizzare la logica applicativa. Tale interazione avviene tramite i meccanismi di Data-Binding e di Dependency Injection: questi permettono di eliminare gran parte del codice che sarebbe scritto usando solo il linguaggio Javascript. E tutto avviene all'interno del browser, rendendo lo sviluppo libero dai problemi tipici del funzionamento crossplatform e indipendente dalla tecnologia server utilizzata.

Si può affermare che AngularJS è ciò che HTML sarebbe stato, se fosse stato progettato per lo sviluppo di applicazioni. Infatti il linguaggio HTML

è linguaggio dichiarativo adatto allo sviluppo di documenti statici e non contiene molto in termini di creazione di applicazioni; di conseguenza lo sviluppo di applicazioni web viene effettuato tramite i seguenti approcci:

- Utilizzo di una library - un insieme di funzioni che sono utili quando si scrivono applicazioni web. Il codice viene caricato e si chiama la libreria quando lo si ritiene opportuno. Ad esempio, jQuery
- Utilizzo di framework - una particolare implementazione di una applicazione web, dove il codice viene impiegato soprattutto nei dettagli. Il framework viene caricato e si chiama il codice necessario quando si ha bisogno di qualcosa di specifico nell'applicazione, ad esempio, Durandal, Tizzone, etc.

AngularJS assume un altro approccio. Si cerca di ridurre al minimo il divario esistente tra il codice HTML statico di una pagina e ciò di cui ha bisogno una applicazione per la creazione di nuovi costrutti HTML. AngularJS si ispira al pattern MVC, come altri framework analoghi quali Knockout o Ember.js. Ma rispetto ai diretti concorrenti, questo framework è in grado di ridurre in maniera considerevole il codice necessario a realizzare applicazioni HTML/JavaScript.

Il Model-View-Controller (MVC), in informatica, è un pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica di business. Di seguito uno schema che ben illustra tale pattern:

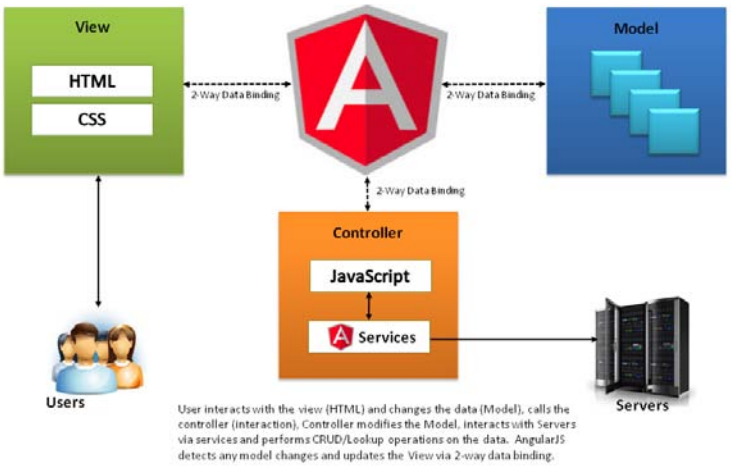

Figura 5.1: *Il pattern MVC usato in AngularJS.*

5.2 Storia

L'idea alla base di AngularJS è frutto dell'ingegno di Miško Hevery e Adam Abrons a cui nel 2009 venne l'idea di implementare un nuovo framework Javascript per semplificare e velocizzare la creazione di Single Page Applications, con il nome di GetAngular, per la gestione end-to-end e che avrebbe permesso ai designer di interagire sia con il front-end sia con il back-end. A seguito dell'assunzione di Hevery a Google, il progetto venne in seguito preso in carico dall'azienda Californiana con il nome di Google Feedback che, apprezzandone lo stato dello sviluppo e le potenzialità, decise di supportarlo esplicitamente lasciandone la guida ad Hevery al quale furono affiancati altri due ingegneri (Igor Minár e Vojta Jína), costituendo di fatto un nuovo team interno a Google (mentre Adam Abros, il secondo creatore del progetto originale, smise di lavorarci) per lo sviluppo del progetto ora chiamato AngularJS. Angular iniziò ad avere una certa importanza quando una compagnia acquisita da Google, DoubleClick, utilizzò il framework per riscrivere parte della propria applicazione. Il successo ottenuto da questo traguardo portò Google ad investire maggiori risorse sul framework, con un conseguente rilascio del progetto qualche anno dopo: infatti la prima versione di AngularJS (la v. 1.0) venne rilasciata nel 2012 ed a questa ne seguirono nuove versioni con una certa frequenza. A partire dalla v.1.2 AngularJS ha abbandonato il supporto per IE 6 e 7 ed a partire dalla 1.3 anche ad IE 8. In questo momento, pertanto, la libreria non offre un supporto realmente cross-browser avendo optato per l'abbandono delle vecchie (ed obsolete) versioni di Internet Explorer. L'ultima versione disponibile, e stabile, di Angular è la 2.0.1, rilasciata il 24 Settembre 2016, e come verrà illustrato più avanti, in questa release sono presenti alcune novità e alcune modifiche sostanziali rispetto alla versione precedente.

5.3 Specifiche

Per poter funzionare al meglio Angular predispone di una serie di componenti per lo sviluppo del codice secondo il pattern MVC:

- Direttive
- Model
- View
- Controller
- Scope
- Data-binding bidirezionale

- Dependency Injection
- Service
- Promise
- Template.

Segue un'analisi concisa di quanto questi componenti rappresentano all'interno del framework:

5.3.1 Direttive

Le direttive sono la funzionalità più unica, predominante e potente disponibile in AngularJS, oltre ad essere una caratteristica specifica per questo framework. Queste permettono di creare componenti HTML personalizzati e riusabili, che possono essere utilizzati per nascondere la complessità della struttura DOM (Document Object Model) e permettono di "decorare" alcuni elementi con comportamenti specifici. Le direttive consentono di estendere il codice HTML permettendo di inventare nuova sintassi HTML, oltre a istruire AngularJS per incorporare le loro funzionalità nella pagina. Le direttive hanno il prefisso "ng-", vengono inserite come attributi HTML e funzionano come elementi standalone. Di seguito le direttive più usate:

- ng-app: Questa direttiva dichiara un elemento come elemento root dell'applicazione e tipicamente viene posizionato all'interno del tag <body> o <html>: <html ng-app> è sufficiente per dichiarare una pagina come applicazione Angular
- ng-bind: Sostituisce il testo contenuto in un componente HTML con il contenuto di un'espressione, e l'aggiorna se l'espressione viene modificata
- ng-controller: Permette di definire una classe controller JavaScript per risolvere le espressioni HTML
- ng-model: E' simile a ng-bind, ad eccezione del fatto che è responsabile del data binding bidirezionale tra lo scope definito nel model e nel view
- ng-repeat: Permette di istanziare un template per item, con cui ciclare sulle variabili di una collection
- ng-class: Questa direttiva permette di caricare dinamicamente gli attributi di classe
- ng-if: Questa direttiva di if dichiarativa è usata per inserire o eliminare un elemento dal DOM, a seconda del valore true o false
- ng-hide e ng-show: Queste direttive consentono di mostrare o nascondere un elemento.

5.3.2 Il Model View Controller

AngularJS incorpora i principi base dietro il pattern design del software MVC originale all'interno del processo di creazione di applicazioni web client-side. Il pattern MVC o Model-View-Controller assume diversi significati in base al contesto utilizzato: per AngularJS non implementa MVC nel senso tradizionale dei framework di sviluppo (come ad esempio Java Spring), ma piuttosto qualcosa di vicino al MVVM/MVW (Model-View-ViewModel / Model-View-Whatever).

Il Model

Il model è semplicemente il dato nell'applicazione, quindi può essere considerato come un normale oggetto JavaScript nella sua definizione classica. Non è necessario che sia ereditato dalle classi del framework, includerlo negli oggetti proxy o usare dei metodi speciali get/set per l'accesso.

Il ViewModel

Il viewmodel è un oggetto che fornisce specifici dati e metodi per il mantenimento di specifici view. Per poterlo definire meglio deve essere introdotto il seguente concetto : nella terminologia di AngularJS uno scope (spesso indicato nel codice con $scope) indica il contesto in cui vengono salvati i dati di un'applicazione (il model) ed in cui vengono valutate le espressioni utilizzate nella view. Il viewmodel è un oggetto $scope che vive con l'applicazione AngularJS: $scope è solo un semplice oggetto JavaScript con delle piccole API progettate per rilevare e trasmettere i cambiamenti al suo stato. Lo $scope è il "collante" tra il view e il controller.

Il Controller Il controller è il responsabile dell'impostazione dello stato iniziale e dell'aumento dello $scope con metodi che ne controllano il comportamento. Vale la pena notare che il controller non salva lo stato e non interagisce con i servizi remoti. Inoltre il controller in Angular è collegato a un elemento del DOM tramite una direttiva.

Il View

Il view è una pagina HTML che viene renderizzata dopo che AngularJS ha analizzato e compilato l'HTML per includere markup e bind inclusi nel codice. Questa divisione crea una base solida per architettare l'applicazione. Lo $scope ha un riferimento ai dati, il controller definisce il comportamento, e il view gestisce il layout e consegna l'interazione al controller per rispondere di conseguenza.

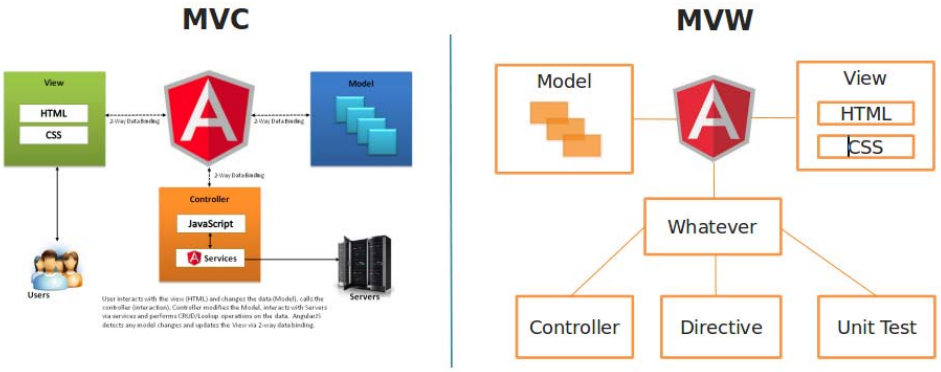

Figura 5.2: *I pattern MVC e MVW utilizzabili in AngularJS.*

5.3.3 Data Binding bidirezionale

Il Data-Binding è la seconda funzionalità più potente di AngularJS dopo le direttive. Permette di eliminare una buona parte di codice inutile lato server per la gestione dei template. Solitamente l'80% del codice di una web application è dedicato alla manipolazione, navigazione e all'ascolto degli eventi DOM. Il data binding di AngularJS nasconde questo codice, permettendo allo sviluppatore di focalizzarsi sugli altri aspetti vitali della web application: in parole povere il data-binding è il meccanismo di sincronizzazione automatica dei dati tra il modello e la view.

Generalmente, in molti sistemi di templating, il data binding è presente solo in una sola direzione, in cui il sistema classico di template unisce template e model alla view. Dopo questo merge, qualsiasi alterazione fatta dall'utente nella view non saranno riportate nel model. Questo significa che poi è lo sviluppatore il responsabile della manipolazione manuale degli elementi e attributi del DOM che devono riflettere questi cambiamenti. Questa è una strada a doppio senso: in una direzione, il modello cambia i valori degli elementi DOM, nell'altra, i cambiamenti dell'elemento DOM, necessitano cambiamenti nel modello. Questo è ulteriormente complicato dall'interazione dell'utente, in quanto è poi lo sviluppatore il responsabile nell'interpretare le interazioni, unirle in un modello e aggiornare la view manipolando manualmente gli elementi e gli attributi del DOM. Quando un utente fa qualsiasi modifica alla view, questo processo diventa sempre più arduo e complicato. Questo a causa del fatto che lo sviluppatore deve interpretare le iterazioni dell'utente, facendo il merge nel model, e aggiornando il model con qualsiasi cambiamento del view. In AngularJS il data-binding funziona in maniera bidirezionale, risparmiando allo sviluppatore l'onere di inserire dati nella view manualmente, sincronizzando automaticamente i dati tra il model e il view e viceversa: ogni modifica al modello dei dati si riflette automaticamente sulla view e ogni modifica alla view viene riportata sul modello dei dati. Per

ottenere questo funzionamento è sufficiente associare il modello allo scope all'interno del controller ed utilizzare la direttiva ng-model nella view . Di seguito un esempio di binding di un valore input ad un tag <H1>.

```
<!doctype html>
<html ng-app>
  <head>
    <script src="http://code.angularjs.org/angular-1.0.0rc10.min.js"></script>
  </head>
  <body>
    <div>
      <label>Nome:</label>
      <input type="text" ng-model="tuoNome" placeholder="Inserisci un nome">
      <hr>
      <h1>Ciao, {{tuoNome}}!</h1>
    </div>
  </body>
</html>
```

Figura 5.3: *Utilizzo del data-binding nel codice HTML.*

Figura 5.4: *Differenze tra il data-binding tradizionale e quello bidirezionale.*

5.3.4 Dependency Injection

La dependency injection consente di combinare insieme componenti allo scopo di strutturare un'applicazione: se all'interno di un componente Angular si necessita di funzionalità offerte da un altro componente è sufficente dichiararne la dipendenza. AngularJS dispone di un sistema di injection integrato che si occupa di creare componenti, caricare dipendenze e di passare questi agli altri componenti se necessario. La Dependency Injection in AngularJS permette agli sviluppatori di dichiarare come la web app è collegata, portando i tradizionali servizi lato server sul client. Questo alleggerisce il lavoro sul server di backend e permette di avere performance migliori. Per avere accesso ai servizi core di AngularJS, è sufficiente aggiungere il servizio richiesto come parametro. AngularJS rileverà automaticamente tale richiesta fornendo un'istanza del servizio.

```
var MyController = function($scope, greeter) {
  // ...
}
MyController.$inject = ['$scope', 'greeter'];
someModule.controller('MyController', MyController);
```

Inoltre, è possibile definire dei servizi personalizzati ed usare l'injection per averli a disposizione.

```
angular.
    module('MyServiceModule', []).
    factory('notify', ['$window', function (win) {
    return function (msg) {
        win.alert(msg);
    };
}]);

function myController(scope, notifyService) {
    scope.callNotify = function (msg) {
        notifyService(msg);
    };
}
```

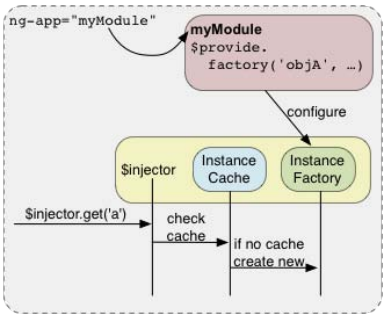

Figura 5.5: *Schema della Dependecy Injection.*

5.3.5 Service

I servizi sono i componenti Angular che offrono funzionalità indipendenti dall'interfaccia utente, consentendo di implementare la logica dell'applicazione, cioé le funzionalità che si occupano di elaborare e/o recuperare i dati (provenienti da un'API REST) da visualizzare sulle view tramite i controller. Un altro ruolo da attribuire ai servizi è quello della condivisione di funzionalità accessibili dagli altri componenti dell'applicazione: controller, filtri, direttive o altri servizi. Per fare un esempio, si supponga di aver bisogno di una funzione per il calcolo del codice fiscale a partire dai dati anagrafici di una persona e che tale funzione venga utilizzata da più componenti dell'applicazione Angular. Il modo migliore di gestire questa esigenza è la sua implementazione come servizio.

5.3.6 Teoria delle "promesse"

Una promessa è un oggetto con un metodo .then(...). Tale metodo ha due parametri in input (opzionali) di tipo function. La firma del metodo then si presenta così:

```
promise.then(onSuccess, onFailure);
```

Entrambe le funzioni, la onSuccess e la onFailure, hanno un parametro. La funzione onSuccess viene chiamata ogni volta che i task asincroni terminano con successo e la funzione onFailure viene chiamata se i task falliscono l'esecuzione. Per ogni promise solo una delle due funzioni può essere chiamata. Il task riesce o fallisce, nessun altro risultato è possibile.

In Angular si ha il servizio $q che fornisce le implementazioni del promise e del deferred . Il deferred rappresenta un task che si concluderà in futuro. Si è in grado di ottenere un nuovo oggetto deferred chiamando la funzione defer() sul servizio $q.

```
var deferred = $q.defer();
```

Inizialmente questo task è in stato di attesa. Alla fine, l'attività viene completata con successo o fallisce. Nel primo caso lo stato di deferred sarà risolto mentre nel secondo caso lo stato sarà respinto. Per valutare lo stato finale del task l'oggetto deferred mette a disposizione due metodi. Il primo metodo resolve(...) viene utilizzato per indicare che l'operazione è riuscita:

```
deferred.resolve('OK');
```

Si noti che il metodo ha anche un parametro: chi chiama il metodo resolve può passare qualsiasi tipo di informazioni utile allo svolgimento del task.

Può essere un tipo semplice come una stringa o un numero o un oggetto complesso. Ad esempio la seguente è una chiamata valida:

```
deferred.resolve(firstName:‘Michael’, lastName:‘Night’);
```

Il secondo metodo reject(...) viene utilizzato per segnalare che l’operazione non è riuscita. Il metodo reject ha anch’esso un parametro, che indica il motivo del fallimento. Anche in questo caso il parametro può essere un tipo semplice come una stringa o un tipo complesso come un oggetto Error :

```
deferred.reject(‘Sorry’);
```

oppure

```
deferred.reject(new Error(‘An Error has been occurred’));
```

Si consideri ad esempio il seguente codice:

```
var promise = deferred.promise;
promise.then(function.result{
alert(‘Success’);
}function(reason){
alert(‘Error’);
});
```

Inoltre, è possibile concatenare più promise, infatti Il valore di ritorno della funzione then(...) di una promise è ancora una promise:

```
promise.then(..)
.then(..)
.then(..);
```

Figura 5.6: *Il funzionamento delle Promise.*

5.3.7 Template

In AngularJS i template sono scritti in HTML, contengono componenti ed attributi specifici del framework, che effettua il merge del template con le informazioni (dal model e dal controller) per visualizzare delle view dinamiche per un utente nel browser. I template HTML vengono analizzati dal browser nel DOM, il quale poi diventa un input per il compilatore di AngularJS, che attraversa il template per renderizzare le istruzioni (le direttive). Collettivamente, le direttive sono responsabili per l'impostazione del data-binding e della view dell'applicazione. E' importante capire che in nessun punto AngularJS manipola il template come una stringa: l'input per AngularJS è un DOM del browser e non una stringa HTML. I data-binding sono una trasformazione del DOM, non una concatenazione delle stringe o cambiamenti dell'innerHTML. Usare il DOM come input, piuttosto che le stringe, è una grandissima differenza che ha AngularJS dai framework simili. Usare il DOM è ciò che permette di estendere il vocabolario delle direttive e costruire le proprie direttive personali o astrarle in componenti riusabili. Di seguito i tipi di attributi ed elementi che AngularJS utilizza:

- Directive: Questo elemento o attributo potenzia un componente DOM esistente o visualizza un elemento DOM riusabile
- Markup: Collega le espressioni agli elementi, utilizzando la doppia graffa {{ }}
- Form Controls: Consente di validare l'input utente
- Filter: Questo elemento formatta i dati da mostrare.

Ecco un esempio dove viene utilizzata la direttiva ng-repeat per ciclare su un array di immagini e popolare ciò che è essenzialmente un template img.

```
function AlbumCtrl($scope) {
  scope.images = [
{"thumbnail":"img/image_01.png", "description":"Image 01 description"},
{"thumbnail":"img/image_02.png", "description":"Image 02 description"},
{"thumbnail":"img/image_03.png", "description":"Image 03 description"},
{"thumbnail":"img/image_04.png", "description":"Image 04 description"},
{"thumbnail":"img/image_05.png", "description":"Image 05 description"}
];
}
```

```
<div ng-controller="AlbumCtrl">
  <ul>
    <li ng-repeat="image in images">
      <img ng-src="{{image.thumbnail}}" alt="{{image.description}}">
    </li>
  </ul>
</div>
```

Figura 5.7: *Un esempio di utilizzo della direttiva ng-repeat.*

In una semplice web application, il template contiene direttive Angular, HTML e CSS in un solo file HTML (tipo index.html). In applicazioni complesse, è possibile mostrare view differenti nella pagina principale utilizzando "partials" che sono segmenti dei template che saranno creati in file HTML differenti.

5.3.8 Testing

Visto che JavaScript è un linguaggio con tipi dinamici che non ha un compilatore in grado di rilevare molti errori, il team di AngularJS ha progettato il proprio framework per facilitare la fase di test. AngularJS contiene moltissime funzionalità per il testing delle applicazioni. Per l'end-to-end testing AngularJS utilizza "Protractor", un end to end test runner che rimuove inutili complessità e si adatta perfettamente al funzionamento di AngularJS, rendendo possibile l'esecuzione di Unit test.

E' inoltre disponibile una estensione per il browser Chrome, chiamata AngularJS Batarang, che se usata permette di rilevare facilmente i problemi di performance e di fare il debug della applicazione tramite browser.

Figura 5.8: *Il logo di Protractor.*

5.4 AngularJS 2, novità e modifiche

Figura 5.9: *Il logo di AngularJS 2.*

Rispetto ad Angular 1.x, il nuovo Angular si presenta diversamente sia sotto l'aspetto estetico che funzionale. La sintassi è completamente nuova e anche internamente il funzionamento è diverso. Buona parte dei concetti e delle funzionalità con cui lo sviluppatore Angular 1.x aveva familiarizzato sono stati eliminati da Angular 2. Di seguito si illustra quali sono i cambiamenti rilevanti introdotti con la nuova versione del framework.

5.4.1 Eliminazione di scope e controller

In Angular 1.x gli scope e i controller avevano un ruolo cruciale nello sviluppo di applicazioni, anche se non sempre il loro utilizzo era appropriato. Quello che era il ruolo di scope e controller ora viene ricoperto interamente dai componenti.

5.4.2 Definizione dei moduli

Anche la precedente versione di Angular incoraggiava un'organizzazione modulare del codice mettendo a disposizione il metodo angular.module(). Nella nuova versione questo metodo non è più supportato e la modularità si appoggia sulle funzionalità standard di ECMAScript 2015.

5.4.3 Two-way data binding

Una delle funzionalità più evidenti di Angular è la gestione del two-way data binding, cioé la capacità di riflettere automaticamente i cambiamenti del modello dei dati sulla view e viceversa. Tuttavia questa capacità talvolta non era necessaria ed un suo abuso portava ad avere un peggioramento delle prestazioni, particolarmente evidenti sui dispositivi mobile: Angular 2 non prevede più il two-way data binding come meccanismo predefinito. Il data binding predefinito è ora unidirezionale, ma è comunque possibile attivare il two-way data binding quando si ritiene sia necessario.

5.4.4 Definizione di direttive

In Angular 2, quelle che erano le definizioni di direttive sono state rimpiazzate utilizzando concetti di ECMAScript 2015 come classi, decoratori e annotazioni. È inoltre cambiata la terminologia permettendo di distinguere tra la creazione di:

- un elemento personalizzato dell'interfaccia grafica (componente)
- un comportamento personalizzato di un elemento esistente del DOM (direttiva).

5.4.5 Rimozione di jqLite

Angular 1.x utilizzava internamente una versione ridotta di jQuery, jqLite, per la manipolazione del DOM. In Angular 2 viene rimossa questa dipendenza preferendo l'accesso diretto al DOM con i metodi nativi ed un conseguente recupero di performance.

5.5 Da Angular 1.x ad Angular 2

L'uscita di Angular 2 apre due scenari per gli sviluppatori che desiderano passare dalla precedente versione alla nuova:

- Creazione di una nuova applicazione Angular 2
- Migrazione di una applicazione Angular 1.x ad Angular 2.

Il primo caso è senza dubbio meno problematico, dal momento che l'obiettivo è la creazione di un'applicazione del tutto nuova. Tuttavia, chi è abituato ad utilizzare Angular 1.x deve ugualmente aggiornare la propria formazione per non cadere nella trappola di considerare la nuova versione del framework come minor release della versione precedente. La soluzione migliore per migrare un'applicazione Angular 1.x esistente verso il nuovo framework sarebbe la sua completa riscrittura. È naturale però che, se questo può essere fatto per applicazioni molto semplici, è del tutto improponibile per applicazioni di una certa complessità. Purtroppo l'iniziale annuncio della mancata compatibilità tra la versione 1.x e la versione 2 del framework ha creato diverse polemiche e ha spinto alcuni sviluppatori ad indirizzare il loro interesse verso altre soluzioni alternative ad Angular. Tuttavia, l'iniziale irrimediabile rottura tra le due versioni di Angular è stata mitigata nel corso dello sviluppo del nuovo framework con un'importante novità. Si tratta di upgrade, noto anche come ngUpgrade, un modulo di Angular 2 che consente la convivenza di Angular 1.x e 2 nello stesso progetto applicativo. In pratica, ngUpgrade consente ad una applicazione Angular 1.x di utilizzare componenti e servizi scritti per Angular 2 e, viceversa, permette di utilizzare direttive Angular 1.x all'interno di un'applicazione Angular 2. Questo facilita sia una migrazione graduale di applicazioni esistenti verso il nuovo framework sia la creazione di nuove applicazioni Angular2 utilizzando componenti già scritti per Angular 1.x.

Capitolo 6

Conclusioni

Nei precedenti capitoli sono state presentate le tecnologie di maggior diffusione per lo sviluppo di siti web responsive: ovviamente queste non sono le uniche, esiste un'ampia scelta di framework, linguaggi e librerie più o meno adatti allo scopo, certo è che tali tecnologie hanno spesso un ciclo di vita direttamente proporzionale alla propria diffusione: come spesso accade nel panorama informatico molti di questi tool appaiono, vengono usati per un lasso di tempo relativamente "breve" ed infine vengono abbandonati. E' per questo motivo che in questa tesi sono state presentate le caratteristiche rilevanti di alcuni strumenti che si possono considerare consolidati: si può dire che HTML5, Bootstrap ed AngularJS sono gli standard de facto per l'implementazione di siti web responsive secondo le ultime tendenze. HTML5 è senza dubbio una delle tecnologie base da conoscere quando si parla di sviluppo web: senza una conoscenza di questo linguaggio è praticamente impossibile poter sviluppare una applicazione web anche molto semplice. Per quanto riguarda Bootstrap e AngularJS, è necessario innanzitutto avere una pur minima conoscenza di HTML, Javascript e CSS per poterne apprezzare le potenzialità: questo però è valido anche per altri tools di sviluppo, e quindi viene ora presentato un confronto critico tra gli strumenti presentati ed i corrispondenti concorrenti di maggiore diffusione.

6.1 Alternative a Bootstrap

Come precedentemente detto, nello scenario dello sviluppo web si presentano molti strumenti di sviluppo: in particolare per Bootstrap il maggiore concorrente è Foundation. Foundation è un framework front-end reattivo costruito da ZURB, una società di product design. Tale prodotto è open-source ed è stato rilasciato nel 2011 sotto licenza MIT. E' modulare e consiste principalmente di fogli di stile Sass (Sass è un meta-linguaggio utilizzato per scrivere un documento CSS in modo pulito e strutturalmente corretto). Si basa su un sistema a griglia di 940px formando un layout reattivo. Foundation for-

nisce agli utenti dei modelli di partenza permettendo loro di iniziare a creare i propri progetti di siti web in maniera molto veloce. Alcune aziende che utilizzano Foundation sono Adobe, eBay, EA, Amazon e Mozilla. Si può dire che Foundation sia una buona alternativa a Bootstrap, ciò nonostante la diffusione di tale strumento è inferiore, come si può notare dal grafico fornito da Google Trends che illustra l'andamento di Bootstrap3, Bootstrap4 e Foundation6 nel periodo tra Agosto 2015 e Agosto 2016 e di seguito riportato:

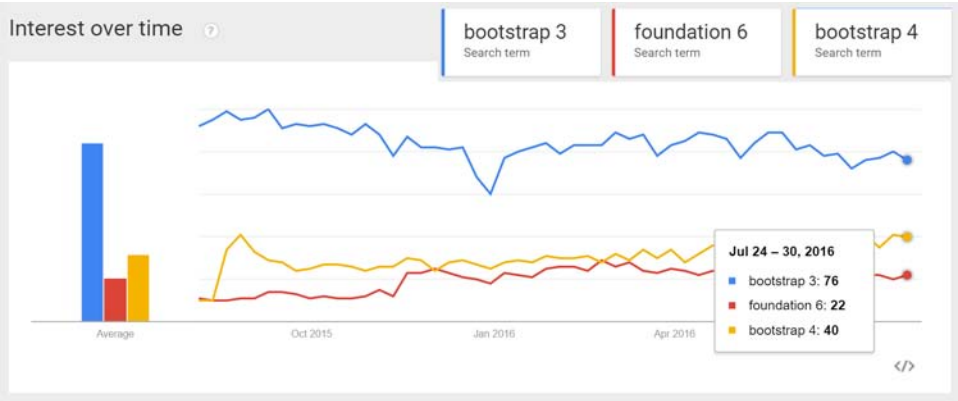

Figura 6.1: *Statistiche dell'utilizzo di Bootstrap e dei competitors.*

Si può notare la marcata differenza dell'utilizzo tra Bootstrap3 e Foundation 6, che si pone poco al di sopra alla diffusione di Bootsrap4, quest'ultimo ancora in fase alpha, quindi non ancora rilasciato in versione stabile. E' plausibile prevedere che questo andamento non cambi nel tempo, anche considerando che pure tra Settembre e Novembre 2016 Google Trends mostra tale andamento quasi invariato, se non per una lieve diminuzione dell'utilizzo di Bootstrap3 accompagnata ad un lieve aumento dell'utilizzo di Bootstrap4, quasi ad anticipare uno switch dell'utilizzo delle due vesioni da parte degli sviluppatori web. Nel contempo Foundation non mostra significative alterazioni nell'andamento della sua diffusione. E' quindi plausibile prevedere che quando Bootstrap 4 sarà disponibile come pacchetto stabile, quest'ultimo prenderà il posto di Bootstrap 3 nella diffusione ed utilizzo del framework per lo sviluppo RWD, mentre Foundation continuerà ad occupare una fetta di mercato, seppur di un certo peso, pressoché invariata.

6.2 Alternative ad AngularJS

Anche AngularJS ha dei forti concorrenti, piuttosto diffusi, che possono essere considerati un'alternativa a questo framework: questi sono EmberJS, Knockout e Backbone. EmberJS attualmente utilizza il motore di template Handlebar, che è un'estensione del popolare motore di template Mustache. Una nuova variante di Handlebar, chiamato HTMLBars è attualmente in

lavorazione. Knockout è una libreria JavaScript che permette la creazione di siti web dotati di una interfaccia utente ricca e dinamica, basata su un sottostante modello di dati. Inoltre Knockout implementa il pattern Model–View–ViewModel (MVVM). È un progetto indipendente e open source sviluppato e mantenuto da Steve Sanderson, un dipendente di Microsoft. Come ha detto l'autore, "(Knockout) continuerà esattamente così com'è e si evolverà in qualunque direzione dove io e la sua comunità di utenti desidereremo portarlo" e, ha sottolineato, "non è un prodotto Microsoft". Backbone.js è una libreria JavaScript con una interfaccia JSON RESTful e si basa sul paradigma di application design Model-View-Presenter (MVP). Backbone è noto per essere leggero, infatti la dipendenza di maggior peso è dovuta ad una libreria JavaScript, Underscore.js, oltre alla libreria jQuery. Backbone è stato progettato in particolare per lo sviluppo di applicazioni web single page. Backbone è stato creato da Jeremy Ashkenas, noto anche per CoffeeScript e Underscore.js. In Figura 6.2 un grafico che illustra l'andamento sull'interesse relativo a questi strumenti nel periodo compreso tra Gennaio e Dicembre 2012, sempre usando Google Trends come fonte.

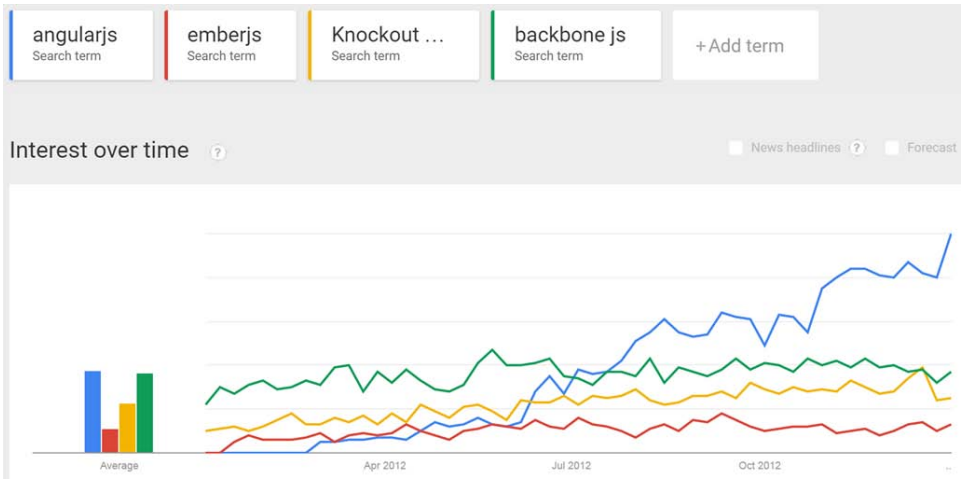

Figura 6.2: *Statistiche dell'utilizzo di AngularJS e dei competitors.*

Come per Bootsrap, anche per AngularJS si può notare l'ampio scostamento tra il framework presentato e i corrispettivi competitors: mentre per Ember, Knockout e Backbone l'andamento dell'utilizzo è stato pressoché lineare, per AngularJS si ha un evidente incremento che lo ha portato, da Luglio 2012, ad essere il framework Javascript più utilizzato per lo sviluppo RWD: è plausibile quindi che questo tipo di andamento continui in tale senso, rendendo AngularJS lo strumento di sviluppo RWD più utilizzato nel panorama web. E' comunque consigliabile monitorare i nuovi prodotti proposti e valutare, sia tramite feedback degli altri utilizzatori che tramite una prova diretta i possibili strumenti che possono emergere nel continuo evolversi

dell'attuale proposta per quanto riguarda lo sviluppo web.

Bibliografia-Sitografia

Bibliografia

- *"Learning Responsive Web Design"* - Clarissa Peterson, ed. O'Reilly
- *"Responsive questions"* - Jeremy Keith
- *"A responsive Design Workflow"* - Tyler Herman
- *"Responsive Design - what it is and how to use it"* - Kyla Nignt
- *"Reinventing HTML"* - Tim-Barnes Lee

Sitografia

- *http://html5doctor.com/html5-forms-input-types/*
- *http://www.antoniosammartino.it/Blog/Default.aspx?id=623*
- *http://www.codelabstudio.it/blog/introduzione-bootstrap/*
- *http://www.designknock.com/tools/resources-tools/useful-html5-css3-responsive-framworks/*
- *http://www.html.it/guide/guida-angularjs/*
- *http://www.html.it/pag/44806/introduzione-a-bootstrap/*
- *http://www.html.it/pag/52589/hello-angular/*
- *http://www.ioeweb.it/lezioni/bootstrap-3-framework-per-la-creazione-veloce-di-siti-web-moderni-parte-1*
- *http://www.mrwebmaster.it/javascript/introduzione_12015.html*
- *http://www.mrwebmaster.it/web-design/introduzione-bootstrap_11392.html*
- *http://www.powerpad.it/c/55835/6764/angularjs–guida-introduttiva.html*
- *http://www.semanticstone.net/tutorial/bootstrap-3-realizziamo-layout-responsivo/*

- *http://www.slideshare.net/roykimtoronto/sharepoint-hosted-addin-with-angularjs-and-bootstrap*
- *https://angular.io/docs/ts/latest/guide/*
- *https://archive.org/web/*
- *https://code.tutsplus.com/it/tutorials/5-awesome-angularjs-features–net-25651*
- *https://docs.angularjs.org/guide/introduction*
- *https://docs.angularjs.org/tutorial*
- *https://en.wikipedia.org/wiki/Backbone.js*
- *https://graygrids.com/best-css-frameworks/*
- *https://it.wikipedia.org/wiki/AngularJS*
- *https://it.wikipedia.org/wiki/Design_responsivo*
- *https://it.wikipedia.org/wiki/HTML5*
- *https://it.wikipedia.org/wiki/Knockout_(web_framework)*
- *https://it.wikipedia.org/wiki/Linguaggio_di_markup*
- *https://www.ibuildings.it/it/blog/sass-il-nuovo-approccio-ai-fogli-di-stile*
- *https://www.keycdn.com/blog/bootstrap-vs-foundation/*