

Corso di Laurea in Informatica Umanistica

Relazione

Il Material design e le nuove tecnologie di programmazione web

Candidato: *Francesco Maridati*

Relatore: *Maria Simi*

Correlatore: *Beatrice Rapisarda*

Anno Accademico 2014-2015

Indice

1. Introduzione progetto. 2
   1. Obiettivi 2
   2. Gli strumenti di lavoro e le fasi di progettazione 3
   3. Argomenti 5
2. Material design 7
   1. Introduzione al Material design 7
   2. Evoluzione grafica e stili 8
   3. Spazio 3D, profondità, ombre, strati 10
   4. Colori, font, immagini, tipografia 13
   5. Animazioni, transizioni, movimenti 18
   6. Alcuni concetti di usabilità 20
3. Web component 23
   1. Progettazione a componenti 23
   2. Custom element 24
   3. Shadow DOM 27
   4. Template 29
   5. HTML Import 30
4. Polymer 33
   1. Architettura di Polymer 33
   2. Elementi personalizzati 34

4.2.1 Uso della libreria 34

4.2.2 Creare un elemento 35

4.3 Data binding e <content> 37

5. Conclusioni 40

6. Riferimenti 41

6.1 Bibliografia 41

6.2 Sitografia 41

**1. Introduzione progetto**

**1.1 Obiettivi**

I dispositivi tecnologici (e la possibilità di connettersi in rete in modo stabile) sono stati da subito utilizzati nel campo della didattica, al fine di permetterne una diffusione capillare della conoscenza e consentire un’accessibilità delle risorse impossibile tramite lo strumento cartaceo. L’apprendimento elettronico, più conosciuto col termine *e-learning*, si è affermato come metodologia di insegnamento e apprendimento che coinvolge sia il prodotto sia il processo formativo (Liscia, 2004). Se col primo termine si intende ogni tipo di materiale multimediale consultabile attraverso il supporto di un dispositivo, il processo formativo equivale alla gestione del percorso didattico offerto. In questa prospettiva, l’apprendimento elettronico necessita di strumenti idonei affinché si riveli tecnica efficace e valido sostegno per lo studente: uno di essi è il *tutorial*, che unisce prodotti e strategie multimediali differenti per semplificare la trasmissione dei suoi contenuti. Le informazioni vengono illustrate nel tutorial tramite video, immagini, registrazioni audio, animazioni e testo scritto, correlati dalla possibilità per l’utente di interagire, al fine di offrire un approccio deduttivo e partecipativo: l’attenzione necessaria è infatti mantenuta attraverso domande a cui rispondere, esempi da completare, piattaforme di lavoro e simili.

Esso permette di offrire una guida completa relativa ad un argomento specifico, il cui contenuto, qualitativamente valido, risponda alle esigenza del lettore e sappia scioglierne dubbi e complessità sollevati dall’argomento.

Nel caso specifico di questo elaborato di laurea, è stato individuato lo strumento del tutorial come mezzo di supporto all’introduzione in ambito accademico dell’argomento trattato, ovvero il Material Design e il modello a componenti per il web. Questi due temi hanno tutte le caratteristiche necessarie per essere trattate in modo più approfondito e costante, ma essendo ancora molto recenti, necessitano di una maggiore diffusione. Il tutorial in questione ha l’onere e lo scopo di ampliare la diffusione di queste nuove tecnologie, specialmente per un pubblico italiano che si vede al momento privo di un’adeguata documentazione a riguardo.

**1.2 Gli strumenti di lavoro e le fasi di progettazione**

Dopo un attento studio dello stato dell’arte, è stato possibile riscontrare la mancanza di documentazione affidabile o scientifica tanto in italiano quanto in inglese sull’argomento del Material design, ma solo qualche articolo di blogger o freelance, eccezion fatta per le *guidelines* dedicate all’argomento sul portale di Google, documento ben redatto e puntuale. Per quanto riguardo il modello a componenti del web (d’ora in avanti associabile al termine inglese Web component), mentre in inglese vi è una notevole documentazione, sia ufficiale che di tipo didattico oppure amatoriale, in italiano presenta ancora diverse lacune. Quindi, salvo prendere come spunti riflessivi alcuni di questi articoli di blog in rete, previa un controllo sulla validità e una successiva citazione, le fonti di studio sono state principalmente i portali on-line ufficiali e i pochi libri inerenti, rigorosamente in lingua inglese. Questa mancanza di documenti non ha però avuto un effetto negativo, bensì di stimolo, rispondendo all’evidente esigenza di avviare un processo di divulgazione anche nella penisola italiana degli argomenti trattati. Essendo il tutorial rivolto ad un pubblico di studenti del corso di laurea in Informatica Umanistica del primo anno, è stato adottato un linguaggio diretto e colloquiale, affinché la mole di informazioni enunciate non risultasse eccessivamente pesante, mentre è stato riservato alla relazione dello stesso un linguaggio più consono e tecnico all’elaborato di laurea. Inoltre, si assume che lo studente disponga di una conoscenza pregressa di base relativa ai linguaggi per la programmazione web quali HTML, CSS e JavaScript, necessari a comprendere fin da subito l’argomento nella sua interezza.

Per favorire l’apprendimento e mantenere viva l’attenzione, è stato fatto un largo uso di immagini ed esempi, sia in forma di codice che propriamente descrittivi, sono state inoltre impiegate animazioni in .gif[[1]](#footnote-1), necessarie, in alcuni casi, a dimostrare in modo inequivocabile, tramite immagini animate, le potenzialità degli effetti visivi proposti dal Material design. Le immagini introdotte nel tutorial son state scelte o per rappresentare propriamente un concetto spiegato (come per esempio la paletta dei colori del Material design o alcuni suoi elementi) oppure per accompagnare il testo a figure riconducibili per deduzione (per esempio, più componenti web che vengono uniti per creare un’applicazione corrispondono all’immagine di più mattoncini della lego impilati per realizzare una costruzione).

Come strumento di supporto alla realizzazione materiale è stato impiegato il pacchetto “HTML5 slide template for Google I/O 2012”[[2]](#footnote-2): un template ideato dal team di Google per le presentazioni in HTML5, integrabile con ulteriore codice necessario, responsive, ovvero adattabile a tutti i dispositivi e relativi schermi, e compatibile con i browser moderni. La formattazione delle slides è stata pensata in strutture differenti in base al loro uso effettivo:

* Presentazione, necessaria a introdurre il tema d’oggetto, su sfondo colorato col nome dell’argomento e il suo eventuale logo;
* Liste, per focalizzare con lo schema delle liste puntate i concetti più importanti nella slide;
* Solo immagini, per mostrare nell’intera slide una figura o un’animazione relativa alla slide precedente;
* Liste e immagini, lista puntata di concetti corredata da immagine inerente;
* Codice, esempi di codice;
* Paragrafi, concetti che necessitano di brevi discorsi per essere espressi.

Tutte le slides, tranne quelle di presentazione, sono costruite secondo lo stesso formato grafico, disponendo i contenuti nella parte centrale della slide, il numero di pagina in basso a destra e una barra orizzontale in alto a sinistra lungo tutta la slide rappresentante uno dei componenti principali del Material design, una *app-bar* monocolore. Essa è provvista dell’effetto ombra tipico dei componenti del Material, di un logo sulla destra e di un titolo per specificare sempre il tema trattato. I tre colori scelti per l’intero progetto sono stati scelti dalla paletta di colore *Cyan[[3]](#footnote-3)* (una tonalità di azzuro) del Material design: il colore primario per la barra orizzontale, il colore secondario per lo sfondo esterno alle slide e l’ultimo, assai chiaro, per lo sfondo delle slide; il font utilizzato per i testi è il *Roboto*.

**1.3 Argomenti**

I temi trattati all’interno del tutorial fanno riferimento a tre macro aree d’argomento: il Material design, i Web component e Polymer.

Il filo comune pensato per collegare queste tre aree è sintetizzabile dentro ai concetti di semplificazione da una parte e di modelli di progettazione e programmazione a componenti dall’altra. Semplificazione perché tutte e tre le tecnologie contribuiscono a fornire ad uno sviluppatore (inteso sia come responsabile della parte grafica che di quella programmatica di un progetto, e per questo ben identificabile con uno studente che deve imparare a muoversi in entrambi gli aspetti) nuovi requisiti e idee atti a migliorare e facilitare la progettazione, la scrittura e la resa del proprio progetto per lo sviluppo web, nonché l’esperienza dell’utente che una volta terminato lo dovrà usare. Modelli di progettazione perché queste tecnologie sono il frutto di attenti studi di squadre di ricerca delle relative aziende, ampiamente testati e innovativi, che forniscono paradigmi d’azione e strumenti per garantire il concetto stesso di semplificazione.

Il Material design è uno stile grafico pensato da Google con cui ha rinnovato l’aspetto estetico delle sue applicazioni; per mezzo di una ricca documentazione online, fatta di dichiarazioni di principi, esempi di elementi già testati e regole di progettazione, Google offre a chiunque la possibilità di applicare questo linguaggio visivo al proprio prodotto, garantendo sia un livello di usabilità e fruibilità delle pagine elevato sia un aspetto estetico gradevole e ordinato. Seguendo o implementando i principi del Material, lo sviluppatore avrà a disposizione valori e proporzioni da applicare ai propri fogli di stile, sistemi di griglie e consigli per adattare i layout, font e colori per ogni esigenza.

Polymer è una libreria framework che offre, tra i vari pacchetti, un’estensione di elementi creati secondo i principi del Material design: questa libreria è costantemente aggiornata e consente di importare in qualsiasi progetto questi componenti funzionanti e facilmente adattabili. Inoltre, mette a disposizione una serie di impostazioni per creare elementi personalizzati, accompagnando e semplificando il lavoro dello sviluppatore con un approccio dichiarativo e ad alto livello. È corretto ricordare che esistono anche altri framework che supportano il Material design, ma Polymer è un progetto di Google, quindi tra le librerie preposte è sicuramente una delle più aggiornate e precise.

Polymer si basa su una tecnologia di sviluppo, chiamata Web component. I Web component sono una collezione di standard emergenti costruiti sul concetto di componenti e riuso. Questo modello sta cambiando il processo di sviluppo delle applicazioni web, favorendo lo sviluppatore nel progettare elementi riutilizzabili ed affidabili, grazie a tecnologie che consentono l’incapsulamento di contenuto non modificabile senza una volontaria intenzione e quindi adattabile in ogni progetto; inoltre, tramite un sistema d’importazione garantisce un facile riutilizzo dell’elemento e un codice più pulito e snello. Tutte funzioni che rendono il Web component un modello attento alla semplificazione del lavoro.

Un tutorial non può certamente sostituire sistemi d’apprendimento consolidati come le lezioni accademiche e un serio studio individuale, ma può essere un valido strumento di approfondimento e chiarimento applicabile a qualsiasi tematica, accessibile a chiunque tramite dispositivo tecnologico, in qualsiasi momento e per quanto tempo ognuno ritiene consono.

**2. Material design**

**2.1 Introduzione al Material design**

Il 25 giugno 2014 Google ha introdotto, durante l’annuale conferenza degli sviluppatori Google I/O a San Francisco, il **Material Design** come una rivoluzionaria esperienza unificata delle **interfacce utente**su dispositivi mobile, siti web e sistemi operativi. L’intento dell’azienda di Mountain View è quello di creare un linguaggio visivo che permetta di progettare un’architettura coerente e funzionale secondo i principi canonici dell’usabilità, leggibilità e navigabilità, insieme ad una grafica moderna volta a rappresentare un aspetto il più realistico possibile. La filosofia del Material Design sta appunto dietro alla metafora della parola Material: essa richiama il concetto di materiale, di un oggetto fisico e quindi tangibile, che abbia una forma specifica, dei bordi, delle ombre, un peso e che reagisca al tocco. Se lo scopo di un’interfaccia utente è quella di facilitare la navigazione, più è intuitiva e familiare, più sarà in grado di raggiungere questo scopo. Così superfici e bordi forniscono indicazioni visive simili alla realtà quotidiana, i movimenti degli elementi sono significativi e coerenti con l’interazione dell’utente permettendo di comprenderne in modo intuitivo ed immediato la funzione, l’utilizzo di colori, font, immagini e icone si basano su attenti studi tipografici che guidano l’occhio prima ancora delle dita, ed ogni elemento corrisponde ad una determinata funzione in qualsiasi dispositivo e per tutte le applicazioni. L’insieme di tutte queste specifiche attenzioni permette di creare un ambiente di navigazione chiaro ed articolato, che fornisce significato a ogni elemento circostante e **«che sintetizza così** i principi classici del buon **design** con le innovazioni e le possibilità offerte dalla **tecnologia** e dalla **scienza»** (Duarte, 2014)[[4]](#footnote-4)***.*** Con il Material Design da una parte Google ha deciso di rinnovare la veste grafica e compositiva di tutti i suoi prodotti, dall’altra lancia ai programmatori di terze parti la proposta di uno standard organizzativo (ben documentato con apposite guidelinesin rete) per fare ordine nel mondo del web design e delle interfacce grafiche ancora troppo caotico e spesso di dubbio gusto, oltre che di difficile comprensione.

**2.2 Evoluzione grafica e stili**

L’evoluzione storica del web design è ascrivibile entro un arco di tempo limitato, proprio come i diversi stili che si sono succeduti. Tralasciando i precedenti e i primi anni del decennio del ’90, dove il concetto di grafica non era argomento di discussione né di investimento di risorse e le pagine web si limitavano a schermi bianchi con pixel monocromatici a rappresentare testo in formato documento, con l’avvento dei browser i layout delle pagine web si incentravano su tabelle statiche composte da righe e colonne. JavaScript nel 1995 nacque come linguaggio per rendere più dinamico il mondo del web ma necessitò di tempo per affermarsi nei browser e avere tempi di caricamento accettabili, mentre Flash nel 1996 introdusse animazioni ed effetti che sono stati abbandonati solo negli ultimi anni. Successivamente nel 1998 vennero lanciati i fogli di stile (detti CSS) per separare in fase di programmazione la presentazione di una pagina dal suo contenuto, non senza difficoltà di supporto dei browser, a dimostrazione del fatto che ancora l’aspetto del prodotto non era considerato strettamente funzionale alla comprensione del contenuto.

Con l’avvento degli anni 2000 si ebbe la vera rivoluzione, in uno scenario in cui la struttura dei siti web si attestava fra quelli appunto costruiti con le tabelle (ora gravemente deprecati) e quelli con i divisori. In questo mare di entità ibride il WEB 2.0 fu il punto di svolta, quando, nel 2004, è avvenuto un sostanziale cambiamento dell’uso stesso della rete a favore dell’utente come fulcro attorno cui costruire la ricerca dei contenuti e l’interazione con gli altri: esempio eclatante il boom mondiale dei Social Media. Da qui un insistente capovolgimento dell’estetica a scapito del contenuto, di animazioni ed elementi fatti per ammaliare, ma che risultavano, oltre che non sempre di gradevole aspetto, molto pesanti da caricare e complessi da capire a un primo utilizzo. Queste sperimentazioni a tratti ingombranti vennero prima ridimensionati con l’ascesa del mobile dove i contenuti e l’esperienza dell’utente dovevano necessariamente essere centrali in dispositivi di piccole dimensioni, e poi migliorati ulteriormente col concetto di *Responsive Web design* per una migliore e fluida adattabilità dei contenuti ad ogni schermo.

Per poter parlare di veri e propri stili grafici di design, bisogna arrivare ad anni più recenti e a grandi colossi del mondo informatico quali la Apple e la Microsoft. La Apple, con l’uscita dei primi dispositivi mobile e poi con i tablet, propose una veste grafica denominata *Scheumorfismo*, mentre la Microsoft con l’aggiornamento del sistema operativo Windows 8 e Windows phone propose il *Flat design.* Lo Scheumorfismo basava il suo stile sulla riproduzione fedele di oggetti e decorazioni reali, come per esempio le sue agende digitali, dove il dettaglio era curato al punto di mostrare le pieghe della carta o la rilegatura in pelle con tanto di cuciture. Il suo design realistico comunicava efficacemente la differenza tra gli oggetti di un sito o di un’applicazione, facilitando notevolmente l’usabilità dell’utente medio e diventando un punto di riferimento per i designers negli anni tra il 2008 e il 2010. Lo Scheumorfismo arricchiva così la veste grafica dei dispositivi in modo profondo, aiutando non solo la comprensione ma anche lo scopo e la familiarizzazione dell’applicazione stessa, unita ad una eleganza e coerenza d’insieme mai vista fino a quel momento (e facilitata dalla novità dell’IPhone stesso). Questo stile però venne gradualmente abbandonato dai suoi stessi sviluppatori nel 2013 quando, con l’aggiornamento di IOS7, anche l’azienda di Cupertino optò per forme più semplici e meno barocche, conscia che la loro creazione era ormai giunta al termine, sia perché il suo processo creativo era lento e macchinoso, difficilmente adattabile alla programmazione in CSS e HTML5 con il relativo Responsive, sia perché, banalmente, il fattore novità si era spento spianando la strada a uno stile più fresco e minimalista dettato dalla Microsoft. Il Flat design proposto da quest’ultima è un approccio basato su geometrie piatte bidimensionali, spazi ben definiti, colori brillanti, riconducibile al De Stijl olandese o al design svizzero degli anni ’50. Elimina gli effetti realistici a favore di un aspetto più essenziale, senza orpelli inutili con l’intento di non disturbare l’utente, focalizzandosi su funzionalità variegate e intense, come nei social network o nei siti e-commerce. Decisamente più adattabile ai vari dispositivi oltre che leggero per il caricamento, specialmente dei siti web, ma anche accusato di essere privo di personalità e di non saper gestire situazioni dove troppe informazioni rischiano di essere confuse dall’estremo minimalismo (TLC, sezione *Flat design e Scheumorfismo)*.

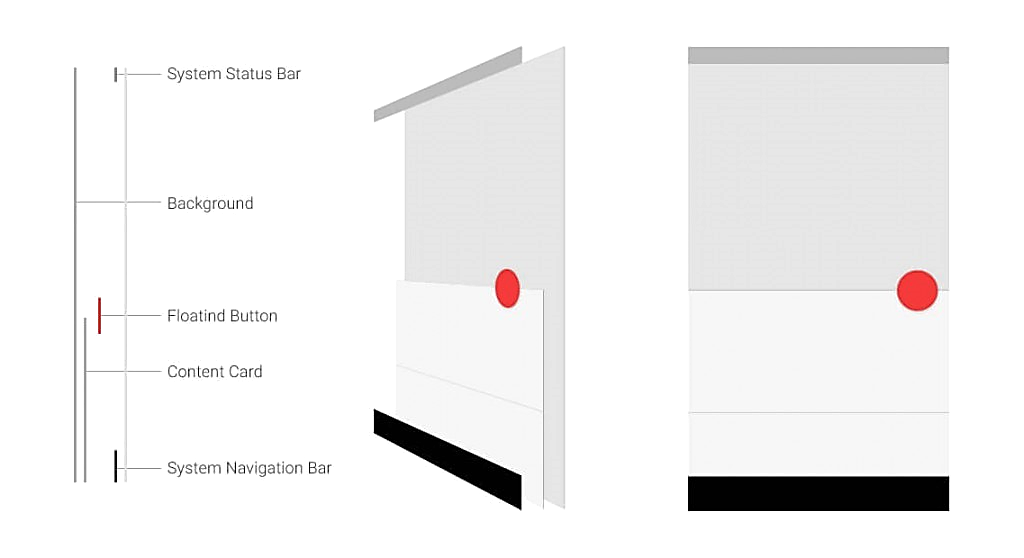
È tra questi ultimi due stili che il Material design si inserisce, coniugando le loro caratteristiche migliori, ovvero l’efficacia di inseguire il realismo scheumorfico con la semplicità e la pulizia del flat, insieme ad uno studio approfondito delle regole di usabilità e una rielaborazione degli elementi fortemente innovativa.

Google ha compiuto un minuzioso lavoro di studio e analisi dietro la nascita del Material Design, che pone le basi per costruire esperienze univoche e coerenti su piattaforme e dispositivi diversi. La documentazione che offre online è estremamente completa e copre una vasta gamma di argomenti in modo molto dettagliato. Non solo specifiche generali riguardo l’uso di ognuno degli elementi, dei colori o dei font, o della funzione delle animazioni, ma anche particolari di una ricercatezza mai impiegata fino ad ora. Ne sono un esempio la lista di valori da usare per i margini e le distanze tra testi (e ancor più fra parole e caratteri stessi), elementi ed immagini all’interno dei layout per ogni dispositivo, le accortezze da seguire per i mancini o per le popolazioni che leggono da destra verso sinistra, finanche alle parole e alle espressioni da usare per rivolgersi agli utenti in modo chiaro. E’ importante sottolineare però che le linee guida sono solo linee guida, indicano la via ma si adattano alle esigenze, e quindi servono come ispirazione e non come regole imposte.

**2.3 Spazio 3D, profondità, ombre, strati**

L’ambiente naturale predisposto al Material design non è composto soltanto dalle due dimensioni canoniche entro cui visualizzare un elemento, cioè l’asse x e l’asse y, ma possiede anche una terza dimensione z, andando a creare uno spazio 3D (Material design, sezione *Environment*). In questo modo ogni oggetto acquisisce uno specifico spessore e di conseguenza un aspetto realistico e una sua forma fisica ben definita. L’asse z è perpendicolare al piano del display e si estende positivamente verso l’utente (fino ad ora nel web la prospettiva veniva modulata dall’asse delle y, mantenendo un ambiente in due dimensioni e quindi poco realistico). Come nel mondo reale, trovandosi per esempio davanti ad una manopola risulta naturale il suo uso girandola o un coltello impugnarlo dal manico, così in un’interfaccia Material, davanti ad un bottone studiato con una precisa proporzione, realizzazione prospettica e proiezione dell’ombra, viene naturale cliccare sopra quello rispetto ad un qualsiasi altro punto dello schermo. Associare un aspetto fisico e una resa realistica ad un elemento aiuta l’utente a dedurne le funzionalità e l’istintiva capacità di riconoscerlo, migliorando così la sua *affordance.* Il concetto di affordance trae origine dalla psicologia della Gestalt degli anni ’20 e ’30, incentrata sui temi della percezione delle cose in relazione agli stimoli e all’esperienza dell’uomo, ma venne coniato dallo psicologo James Jerome Gibson nel 1979. Secondo Gibson l’affordance di un oggetto è ciò che permette a chi lo osserva di capire in modo immediato il suo uso e le sue finalità intrinseche, cogliendo la possibilità di un’azione con esso. L’affordance non è legato tanto all’oggetto in sé, quanto al contesto in cui è inserito e che permette di comprenderlo. Conseguentemente questo concetto ha avuto un’importante diffusione nel mondo del design grazie agli studi dello psicologo Donald Norman, uno dei padri della moderna psicologia cognitivista, il quale ha approfondito anche la sua applicazione nel campo delle interfacce digitali. In tal senso è più appropriato parlare di affordance percepita più che assoluta, relativa cioè alle operazioni che un utente coglie (percepisce appunto) in funzione delle quali vengono attivate specifiche funzioni. Quello che il Material design si prefigge è il connubio tra l’affordance e l’estetica dei suoi componenti, sviluppati secondo un equilibrato uso prospettico (grazie a forme, ombre, spessori), di colori e di movimenti associati (Giannini, *Design,* p. 63).

Le proprietà fisiche degli elementi del Material possono variare nelle dimensioni dell’asse x e y, ma rimangono costanti nello spessore, tenendo il valore dell’asse z fisso a 1dp: ciò permette di avere un ambiente coerente ed equilibrato, in linea con il concetto di *quantum paper* proposto dal team di Google. Di fatto avviene una semplificazione dei componenti scelti entro un gioco di linee e forme, trasformandoli in superfici digitali e tangibili, pur mantenendo una sensibile profondità dettata da realistiche ombre conseguenti alla disposizione nei vari livelli di produzione uno sopra all’altro. L’asse z fornisce l’effetto di profondità ai vari elementi, ma la sua funzione non è tanto fornire una prospettiva quanto una stratificazione: in questo modo la struttura grafica assume un sistema di gerarchie; effetto accentuato anche dall’uso di colori vivaci su aree bianche ed ombreggiate, che sfrutta il contrasto fra colori tenui e colori decisi. La profondità di questi livelli rappresenta la distanza che essi hanno sull’asse z, l’unità di misura è la *density-indipendent pixel* e poiché tutti gli elementi hanno lo stesso spessore di 1dp, l’elevazione corrisponde alla distanza della parte inferiore di uno con la parte superiore dell’altro. La suddivisione di una pagina in più livelli migliora l’esperienza dell’utente premettendo una più immediata focalizzazione e comprensione dell’uso di queste superfici.



Le diverse superfici vengo quindi posizionate ad altezze differenti (e in base alla posizione generano una determina ombra) così da porre in primo piano gli elementi più importanti. Tra i componenti principali, all’altezza di 2dp vengono poste le *card* (i blocchi primari dell’interfaccia) con i contenuti della pagina, il *FAB* (il famoso bottone tondo) a 6dp, il *menu* a 8dp e il *nav drawer* a 16dp. Tutti i componenti del Material hanno una posizione di default (come quelle citate sopra) e una che si attiva all’interazione dell’utente: passando il cursore sopra a un elemento o toccandolo su mobile, questo si attiva e reagisce all’impulso focalizzando l’attenzione su di esso, per poi tornare nella posizione di riposo una volta conclusa la funzione. Tutto ciò mantenendo i canoni di coerenza del mondo fisico per cui gli oggetti si possono sovrapporre o incollare tra loro ma non possono passare attraverso, oltre al fatto che possono proiettare ombre e riflettere la luce. Tutti gli oggetti rientrano in una relazione gerarchica padre-figlio, dove l’elemento figlio è subordinato all’elemento padre, di conseguenza l’interazione fra gli oggetti è determinata dai vincoli di gerarchia. Gli elementi figli sono posti ad una certa distanza dall’elemento padre e non può inserirsi un terzo elemento fra i due, mente gli elementi fratelli hanno stessa profondità quindi rispondono allo stesso flusso d’azione. Alcune eccezioni si possono trovare con i FAB che sono indipendenti e non seguono il flusso delle azioni rimanendo fissi nella loro posizione o il *dialog,* una sorta di *alert* moderno che compare letteralmente dal nulla e ha massima profondità fissata a 24dp. L’importanza e la gerarchia dei contenuti che un programmatore vuol dare al proprio progetto determinano la profondità degli oggetti (Material design, sezione *Elevation and shadows*).

Il Material design si professa uno stile completamente responsive, adattandosi in modo dinamico e autonomo ad ogni dispositivo dal mobile, al computer, alla TV fin anche ai più moderni orologi digitali, senza perdita di informazioni e contenuti sempre ridimensionati in modo proporzionale alle dimensioni dello schermo.

Le superfici possono essere create o distrutte, possono cambiare forma, frammentarsi o ricongiungersi, una sezione della superficie può sollevarsi ed allargarsi (per esempio quando si clicca un elemento di una lista che si espande allargando il suo dettaglio nello schermo). Queste proprietà aiutano a creare applicazioni immersive che rispondo alle interazione anziché saltare in stati diversi o in nuove pagine, mantenendo continuità in un unico ambiente.

Per accentuare al massimo l’intuitività, ad ogni oggetto è associato un certo comportamento: per esempio i FAB sono sempre tondi, in primo piano e incentivano il tocco specialmente nei dispositivi mobile, mentre cliccando l’icona in alto a sinistra della *tool-bar* l’utente si aspetterà sempre la fuoriuscita del menu di navigazione.

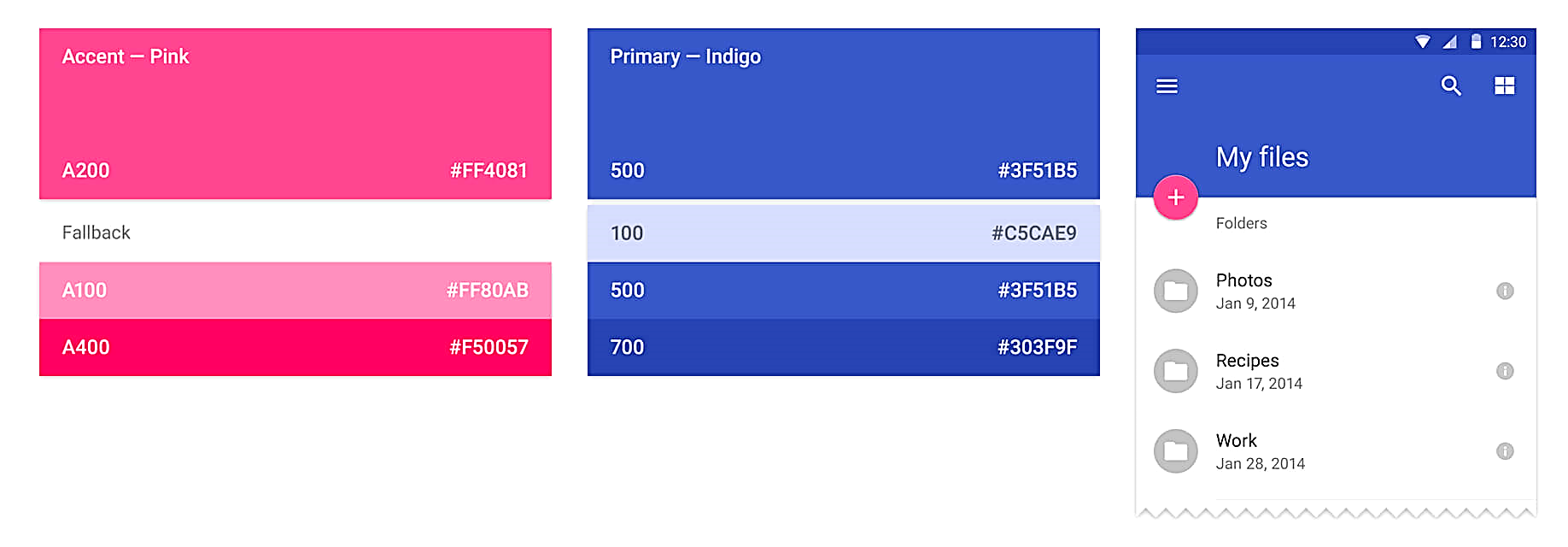
**2.4 Colori, font, immagini, tipografia**

A sostegno della metafora fisica del Material design, un forte contributo è dato da ciò che appare per mezzo di questo inchiostro digitale. Google definisce ogni caratteristica nel minimo dettaglio offrendo risorse e modelli da consultare per i programmatori.

Forme e strutture, griglie e arrangiamenti aiutano in modo elementare l’utente nell’uso dell’interfaccia, ma anche il colore e l’aspetto possono esprimere utili informazioni. Il colore infatti può essere usato per dirigere l’attenzione o implicare gerarchie e strutture. Il Material invita all’uso propositivo di colori audaci, sia per una questione estetica che per semplificare la vita all’utente.

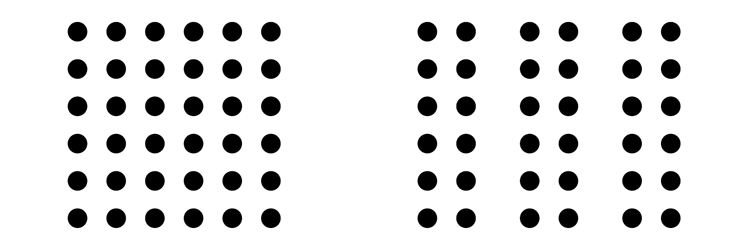
Dominano i colori vivaci in contrasto con ambienti tenui, ombre marcate e luci brillanti per accentuare la profondità. È raccomandata la scelta di colori primari per i blocchi più grossi come la barra degli strumenti per essere immediatamente riconosciuti (chiunque associa subito siti come facebook o youtube con il loro colore predominante). Google offre una vasta gamma di scelta dove per ogni colore primario si possono scegliere diverse tonalità di esso per personalizzare la propria tavolozza. Successivamente va scelto un colore accentato, luminoso, più saturo per richiamare l’attenzione su certi elementi come il FAB o per enfatizzare alcuni testi. Per le aree complementari e in secondo piano, ogni colore primario ha a disposizione una paletta di sfumature da cui attingere, ad ognuna delle quali è assegnato un valore numerico crescente dalle tonalità più chiare a quelle più scure; i colori accompagnati da una lettera indicano tinte più sature mentre al colore primario Google associa il numero 500. La raccomandazione è di attenersi a massimo tre tonalità del colore primario e a un colore accentato per mantenere una tavolozza semplice ma versatile. Considerando che la maggior parte delle volte il colore accentato viene usato vicino o sovrapposto al colore primario, è importante essere sicuri di creare il giusto contrasto fra i due per evitare di perdere la potenza immersiva che potrebbe avere. Il colore deve servire a dichiarare a colpo d’occhio ciò che si vuole esprimere (Material design, sezione *Style*).

Secondo il modello di Treisman e Gelade (1980) riguardo i meccanismi della ricerca visiva, lo stadio mentale che rileva ed analizza simultaneamente le caratteristiche degli oggetti (colore, forma, orientamento, posizione ecc.) viene definito *preattentivo* (mentre quello attentivo li combina creando così la percezione reale dell’oggetto). Questo processo visivo tende inconsciamente a fare ordine nell’insieme di stimoli visivi, organizzando gli elementi visuali in gruppi per formare immagini unitarie e organizzate. Per identificare sulla base di una caratteristica (per esempio il colore) un determinato stimolo in uno scenario d’insieme, il sistema cognitivo dell’uomo si attiva a livello preattentivo: questa caratteristica risalterà visivamente rispetto alle caratteristiche degli altri stimoli (definiti distrattori). In una pagina web in cui è ben netta la differenza di rapporto fra distrattori e stimolo, come nell’uso tra colore primario e colore accentato associato a un certo bottone o contenuto che si vuole mettere in risalto, il tempo di focalizzazione sarà immediato ed efficace evitando rallentamenti e un carico eccessivo di analisi del contenuto. All’aumentare dei distrattori, aumenta il tempo impiegato per la ricerca e la lettura (Giannini, *Design*). È importante quindi che il particolare su che si vuol far focalizzare sia o isolato e in netto contrasto (come un FAB o una card elevata rispetto alle altre) o che catturi da solo l’attenzione con un’animazione adeguata o un uso ordinato del colore.



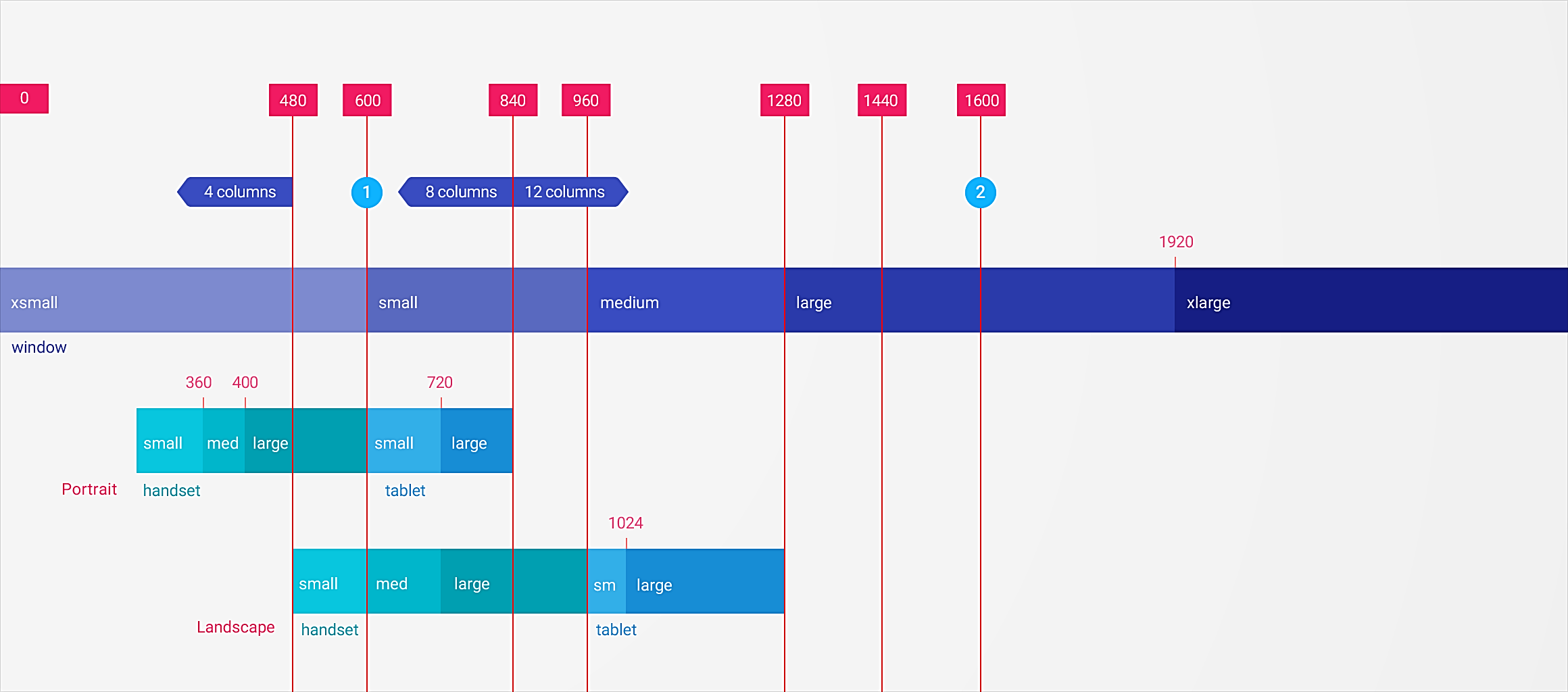
*Roboto* è il vecchio font sens serif di Android rivisitato per il Material Design e scelto come compromesso migliore di leggibilità grazie al suo carattere tondeggiante e bilanciato che favorisce una più piacevole lettura del testo a schermo. Esso possiede 6 pesi: sottile, leggero, regolare, medio, grassetto e nero. Avere queste varianti disponibili in aggiunta alla dimensione e al colore permette di comunicare in modo chiaro e di creare una gerarchia nel testo. L’unità di misura del font è lo scale-indipendent pixel. Per motivi di accessibilità gli utenti possono personalizzare a loro piacimento la dimensione dei caratteri. I font sens-serif sono comuni nel mondo della carta stampata ad esempio nei libri e sono considerati più confortevoli alla lettura di lunghi testi, così come dalla tipografia viene ripresa la buona norma di limitare i font a uno solo per non perdere la continuità e non rendere confusionario l’ambiente di lettura. Il team di Google consiglia il font Roboto per le lingue di origine latina, greca e cirillica, mentre per le lingue cosiddette a linea alta come l’indiano o l’arabo, e quelle dense come il cinese o il giapponese, propongono l’uso del font Noto, più adattabile alle loro esigenze.

Secondo il principio di vicinanza proposto dalla teoria della Gestalt sulle leggi fondamentali della percezione visiva, la distanza relativa fra gli elementi influenza la percezione con cui l’uomo li organizza in gruppi, quindi gli elementi più vicini vengono riconosciuti come logicamente collegati.



Nella progettazione di un sito web, gli elementi dell’interfaccia che sono logicamente connessi devono essere raggruppati utilizzando il principio di vicinanza. Questo perché il raggruppamento permette di ridurre le complessità e il carico cognitivo dell’utente. Il Material design utilizza questa legge per la composizione dei layout dentro schemi a griglie o la disposizione visuale delle liste di elementi, insieme ad un uso equilibrato dello spazio bianco per evitare raggruppamenti percettivi indesiderati. Il principio di somiglianza invece afferma che gli elementi simili per colore o forma sono percepiti anch’essi come correlati. In una scala di valori, l’aspetto del colore è quello con l’impatto più immediato perché percepito istantaneamente, specie se utilizzato in forte contrasto (utile per chi soffre di daltonismo) come propone il Material design e tenendo conto che un uso eccessivo di colori perde di efficacia in fase di raggruppamento. Mentre tra forma e dimensione, il Material design sfrutta più quest’ultima sia perché propone principalmente forme semplici e quindi inefficaci per il principio di similarità, sia perché ha fatto dell’adattamento ai diversi dispositivi e schermi la sua forza per una resa dei contenuti in dimensioni sempre appropriate (F2.net, sezione *I principi Gestalt: Proximity*).

Affinché l’aspetto risulti ordinato è necessario definire dei valori e dei canoni entro i quali collocare tutti gli elementi: il sistema suggerito è quello delle griglie con valori ottimali per l’inserimento del testo, delle icone e delle immagini. Questa architettura dell’interfaccia è un particolare che salta subito all’occhio se non viene rispettato, rendendo caotico e poco gradevole l’aspetto complessivo, per questo Google si è impegnato a fornire distanze e valori chiave da rispettare per far risultare tutto in ordine. Il sistema delle griglie crea dei confini nel layout che garantiscono un posizionamento familiare e coerente di contenuto e articoli, creando una sensazione coesa e ordinata del contesto. Per adattarsi ad ogni schermo, fornisce un sistema di breakpoint a cui associa un certo numero tipo di griglie a colonne per configurare il layout:

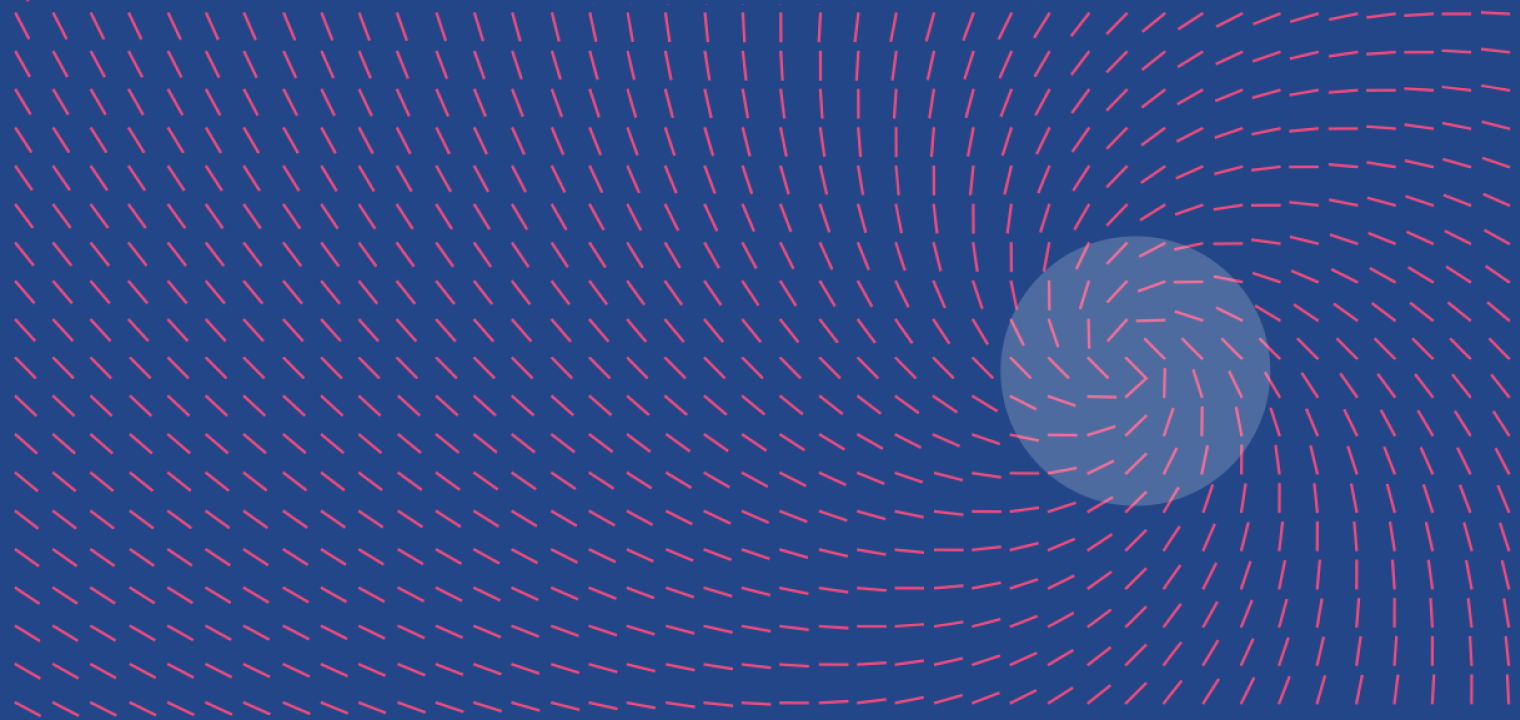


Un buon linguaggio, che sia verbale o visivo, crea delle buone connessioni col soggetto, per esempio delle buone immagini possono richiamare sensazioni inconsce e familiari. Un’immagine deve trasmettere informazione, personalizzare l’esperienza utente ma in modo appropriato al contesto, oltre che coinvolgerlo. La scelta di un messaggio visivo ricade tra immagini fotografiche, illustrazioni o icone. Adoperare un’immagine fotografica aiuta a sottolineare l’informazione relativa ad una persona, oggetto o luogo, avendo l’accortezza di usare immagini di qualità e centrando il punto focale per un effetto immediato. Esse devono essere dinamiche e legate al contesto, evitando filtri ed eccessive modifiche, oltre che composizioni finte e innaturali. In un ambiente astratto invece l’obiettivo è la chiarezza senza eccessive decorazioni, usando una prospettiva coerente, una paletta di colori appropriata e una trama consistente atte a mantenere una specifica connessione. Infine le icone inseguono una simbologia e una chiarezza ancora più ricercata, spiegando sinteticamente il significato. Devono essere facili da identificare e interpretare, per questo vengono usate nelle attività più comuni come la ricerca o la condivisione. Il Material design fornisce una vasta gamma di icone da sfruttare, oltre che una dettagliata guida di istruzioni per crearne di nuove. La scelta di ognuna di queste tre forme visive dipende dall’equilibrata dipendenza tra contenuto e messaggio del proprio lavoro, dalle emozioni che si vogliono comunicare oltre che dal gusto personale.

**2.5 Movimenti, animazioni, transizioni**

Nel mondo reale gli oggetti non appaiono o scompaiono dal nulla, ma si muovo nello spazio seguendo i principi della fisica. Ogni animazione deve essere in grado di mostrare in modo immediato gli elementi importanti della schermata e deve produrre effetti attesi, senza disorientare o rendere effetti belli alla vista ma poco pratici. Le animazioni sono sensibili all’input dell’utente e devono essere la conseguenza del suo gesto: anche in questo caso si parla di metafora materiale in quanto il movimento deve richiamare il comportamento degli oggetti del mondo reale in modo elegante e semplice. Le animazioni del Material design si sposano con il principio di *destino comune* proposto dalla teoria di Gestalt, per cui gli elementi che si muovono simultaneamente sono percepiti come correlati. Quando un movimento è contemporaneo e segue una direzione unitaria, questo restituisce un equilibrio d’insieme e di coerenza in cui è naturale percepire una correlazione di causa-effetto tra il punto di partenza che ha scatenato il movimento e il punto d’arrivo nella visualizzazione dell’evento.

Ci sono due forme principali di feedback visivo in risposta al tatto, ovvero l’effetto radiale e l’effetto d’avvicinamento. Il primo è un sottile effetto a catena, che si irradia verso l’esterno dal punto di contatto fino ai limiti della superficie corrispondente, mentre per gli elementi senza bordo l’effetto a catena si estende solo quanto basta per dare un senso di dimensione circolare dell’elemento. L’effetto viene chiamato anche “ad inchiostro” perché è paragonato ad una goccia di inchiostro che cadendo su di un foglio di carta si diffonde fra le sue trame.



Questa idea di movimento radiale che si espande dal tocco col dito o col mouse è la chiave del tema del Material design. Le animazioni sono provocate dall’utente tramite il tocco col dito, la voce, il mouse o la tastiera a seconda del dispositivo utilizzato. All’attivazione dell’evento in input il sistema fornisce un’immediata reazione visiva che parte dal punto di contatto, indicando dove e quando si verifica e che è stato correttamente ricevuto: il meccanismo alla base è chiamato *Touch Ripple* e provoca l’effetto visivo di propagazione radiale verso l’esterno dal punto di immissione, di cui modula la durata dell’evento, oltre che la direzione e l’effetto visivo dall’epicentro. Quest’effetto agisce solo sulla superficie dell’elemento selezionato. Una conseguenza a tale azione consiste nel rendere l’oggetto reattivo, facendolo sollevare al tatto e ponendolo in uno stato attivo, come se fosse magneticamente attratto dal dito o dal mouse. La sua forma può cambiare, può esserne generata una nuova, può esserne modificata una parte, trascinata o cancellata, per poi tornare allo stato di riposo in modo lineare o radiale. Quando si scatena la creazione di un nuovo elemento, esso deve generarsi a partire dal punto in cui l’utente genera tale elemento senza rompere la relazione tra i due.

I movimenti devono essere funzionali a direzionare senza problemi gli utenti tra i contenuti, a spiegare i cambiamenti nella disposizione degli elementi e a rafforzare le gerarchie degli elementi. La transizione deve essere chiara e pulita, oltre che indicare istintivamente all’utente dove concentrare la propria attenzione. Il concetto di continuità visiva è alla base di queste transizioni. I tempi di transizione invece devono essere coerenti e gli elementi coordinati tra loro (Material design, sezione *Animation*).

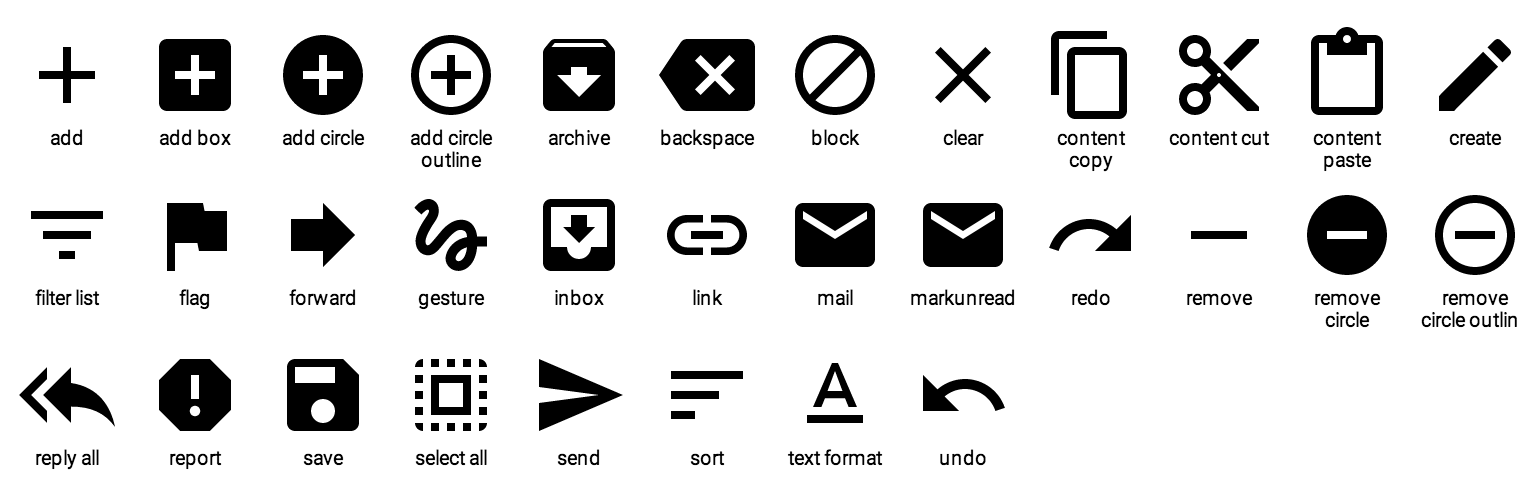
Il movimento deve aiutare l’utente a orientarsi e fornire spunti su come funziona, deve essere utilizzato volutamente e con giudizio. Il tempo migliore stimato è sui 300 millisecondi circa, né troppo lento né troppo veloce. L’animazione, se usata con criterio, può essere un’ottima personalizzazione della propria applicazione o del proprio sito web, donandogli letteralmente vita, fatto che crea una buona interazione con l’utente facendolo sentire in controllo dell’applicazione. Una buona animazione grafica facilita la comprensione della forma concreta di un oggetto e di come manipolarlo. Dal movimento degli oggetti possiamo dedurre se questi siano leggeri o pesanti, flessibili o rigidi, piccoli o grandi. L’utilizzo basilare delle animazioni grafiche è nelle transizioni; ma un’applicazione diventa davvero sorprendente e innovativa per l’utente quando gli effetti di animazione sono utilizzati in maniera creativa oltre che funzionale (MDB, sezione *The ultimate guide*).

**2.6 Alcuni concetti di usabilità**

Secondo un’analisi condotta sull’attenzione sbrigativa e sommaria che l’utente mediamente applica alla propria navigazione nel web, Steve Krug, esperto di usabilità web, identifica cinque importanti principi da seguire per cercare di trattenere e coinvolgere l’utente:

* Sfruttare le convenzioni
* Creare una gerarchia visiva
* Identificare aree ben definite
* Rendere ovvio ciò che è cliccabile
* Minimizzare le distrazioni

Analizzandoli uno ad uno è possibile scorgere una certa attinenza con le caratteristiche proposte dal Material design. Le convenzioni sono delle norme accettate in un determinato ambito che facilita la comprensione e l’intesa. Un esempio di convenzione sono i cartelli stradali, con forme, colori, scritte e posizioni riconosciute da tutti i guidatori. Così anche il mondo del web ha negli anni adottato degli standard funzionali, come la posizione del logo e dei menu nella pagina (sempre nella parte alta dello schermo) o i sistemi di pagamento per un’applicazione commerciale. Anche il Material design fa largo uso delle convenzioni, fornendo ad ogni componente specifiche funzioni o movimenti che ricorrono uguali ogni volta che vengono impiegati in un progetto o ricorrendo a numerose icone per semplificare e associare un concetto ad un’immagine stilizzata.



Una maniera per rendere una pagina comprensibile è che vi sia una chiara gerarchia visiva data dalla cura delle relazioni e correlazioni di importanza fra contenuti, sia nel testo (titolo, sottotitolo, articolo) che nel layout (raggruppamento di argomenti, nidificazione). Il Material design fonda la sua resa visiva su questo concetto, assegnando profondità, gerarchie e scorrimenti su piani il compito di tenere sempre focalizzata l’attenzione sul contenuto di una pagina.

Scomporre la pagina in aree ben definite permette all’utente in modo immediato di scegliere a quali aree dedicarsi e quali ignorare. Grazie alle animazioni immersive di certi elementi del Material, questo concetto è stato esasperato a tal punto da poter disporre con un semplice tocco il contenuto dell’oggetto interessato sopra all’intero ambiente abbattendo il superfluo.

Un importante elemento di usabilità e chiarezza di navigazione è il poter identificare senza possibilità d’errore un bottone per inviare una richiesta o un link per condurre una ricerca in un altro sito. Il Material design agisce in questa direzione collocando in diversi stati di attività i bottoni (posizionando un bottone a un livello più basso se inerte, intermedio se attivo, elevato se cliccato) e lanciando il FAB, acronimo di flating action button (tasto d’azione flottante) per promuovere un’azione; esso si distingue da tutti gli altri per la sua forma circolare, la sua posizione fissa e la presenza di un icona al suo interno per definire in modo immediato il suo funzionamento.

Sempre secondo Krug uno degli errori più ricorrenti è il sovraffollamento di informazioni in una singola pagina che affatica l’utente e lo confonde in un insieme incontrollato di contenuti e distrazioni superfluo. Confrontando le interfacce delle applicazioni di Google precedenti allo stile del Material design, si nota un incrementale sfoltimento di testi ed elementi, in linea con il nuovo atteggiamento minimalista ed ordinato, disegnato dentro griglie proporzionate (Krug, *Don’t make me think*, cap. 3).

La psicologia cognitiva applicata al mondo del design ha sempre sposato, in tutti i campi, l’importanza della funzionalità, dell’usabilità, della forma e della funzione. Quando questi principi vengono affiancati al piacere d’uso di un certo prodotto, o addirittura ad una partecipazione emotiva, questo assume un valore maggiore favorendo ulteriormente l’esperienza dell’utente. Infatti l’utilizzo di un dispositivo risulta più immediato e semplice se viene considerato bello e piacevole da parte dell’utente, dimostrando una diretta relazione tra estetica ed usabilità (Giannini, 2011).

**3. Web component**

**3.1 progettazione a componenti**

I Web Components sono una collezione di norme emergenti prodotte dagli ingegneri di Google che vanno nella direzione del w3c e che permettono di creare componenti dell’interfaccia utente riutilizzabili nei documenti e nelle applicazioni web. Quello che di concreto offrono è di estendere il vocabolario per l’HTML con nuovi tag realizzabili da chiunque o di modificare quelli esistenti con nuovi attributi e funzioni. L’intenzione è quella di applicare anche al mondo della programmazione web quei principi di suddivisione in componenti avviata in contesti come le architetture software (sviluppo del software basato su componenti). Il modello a componenti consente l’incapsulamento e l’interoperabilità di elementi individuali impacchettando frammenti di codice HTML, CSS e JavaScript. Quindi, i Web component lasciano a chiunque la possibilità di creare e usare i propri elementi che appaiono uguali nell’aspetto e nelle funzioni a quelli standard (approvati da tempo dal w3c) ma che possono essere usati senza la lunga trafila di approvazione dell’intero settore. I Web component infatti sono basati su un nuovo set di specifiche che necessitano ancora di essere adottati ufficialmente. Da sempre una novità per diventare uno standard ed essere approvata dal consorzio del w3c richiede procedure lunghissime, ma questa nuova tecnologia riesce ad eludere tutti i processi e a funzionare già su tutti i browser di ogni dispositivo. Sebbene il concetto di Web Component sia stato introdotto alla *Fronteers Conference 2011* da Alex Russell durante la sessione “Web Components and Model Driven Views” tenuta ad Amsterdam[[5]](#footnote-5), ad oggi solo Chrome e Opera sono i browser che adottano nativamente questa tecnologia. Per ovviare a questo inconveniente è necessario l’ausilio di polyfills, ovvero dei codici JavaScript scaricabili che forniscono dei servizi aggiuntivi rendendo universale la compatibilità dei nuovi elementi fra i browser (come successo all’uscita di alcuni elementi HTML5). Google ha creato una libreria di polyfills chiamata webcomponents.js da richiamare nella <header> di un progetto per garantire la corretta lettura dei nuovi tag HTML, scaricabile dal sito dedicato [www.webcomponents.org](http://www.webcomponents.org)o da qualunque piattaforma di hosting come *GitHub.* Finché però tutte le aziende leader nel settore della navigazione web non sposeranno a pieno la filosofia e i principi che stanno alla base di questa nuova tecnologia, la comunità dei Web component non potrà mai espandersi a dovere, impedendo lo sviluppo di uno standard che potrebbe apportare sostanziali migliorie al mondo della programmazione web e mobile. Il progetto è ambizioso: l’innovazione che propone questo modello di progettazione è da una parte aumentare, personalizzandolo, il valore semantico di un elemento HTML, dall’altra di semplificare la sua sintassi e di ridurre la complessità dei documenti ospitanti (come l’abuso dell’incasellamento con i tag div) importando pacchetti da file esterni: così facendo ad un singolo elemento dell’HTML corrispondono degli eventi creati appositamente che agiscono sul tag (senza bisogno di capire necessariamente le dinamiche che stanno dietro) e lo restituiscono a video arricchito delle specifiche implementate. I Web components sono composti da quattro tecnologie principali che permettono tutto questo: i Custom elements, lo Shadow DOM, l’HTML Import e il Template (webcomponents, sezione *Polyfills*).

**3.2. CUSTOM ELEMENT**

I Custom elements (la cui traduzione corrisponde a elementi personalizzati) sono la chiave per la creazione dei nuovi tag, nati con l’esigenza di ampliare il vocabolario di elementi del linguaggio HTML. Con essi è possibile definire e registrare nuovi elementi per il DOM, ereditare ed estendere quelli esistenti e raggruppare insieme in un unico tag funzionalità personalizzate. I tag per definire un Custom element devono contenere necessariamente un trattino (-) per distinguere gli elementi personalizzati da quelli standard, ad esempio <x-element> o <custom-div>. Le fasi di sviluppo che danno vita ad un elemento personalizzato sono:

* Creazione
* Dichiarazione delle proprietà
* Definizione di metodi del ciclo di vita
* Registrazione
* Estensione

Una volta creato l’elemento, esso è riutilizzabile in qualsiasi applicazione.

La prima fase per lo sviluppo di un Custom element avviene invocando il metodo Object.*create(),* che crea un oggetto JavaScript in cui definire metodi e proprietà. A questo metodo vengono passati due parametri, il primo è *HTMLElement.prototype,* indispensabile per settare i metodi e le proprietà standard disponibili, il secondo, opzionale, che contiene le proprietà per il nuovo oggetto. Il nuovo oggetto viene contemporaneamente salvato in una variabile.

var objectPrototype = Object.create (HTMLElement.prototype);

Una volta definito il prototipo dell’elemento, questo assume le sembianze di un contenitore in cui è possibile definire le proprietà dell’oggetto, dichiarabile tramite il metodo *Object.defineProperty* per creare una singola proprietà, o il metodo *Object.defineProperties* per creare più proprietà.

var objectPrototype = Object.create(HTMLElement.prototype);

Object.defineProperty(objectPrototype, ‘title’, {

writable: true

});

Nell’esempio, dopo la creazione dell’oggetto viene invocata la funzione per definire una singola proprietà, a cui viene passato come primo parametro il nome dell’oggetto, come secondo il nome della proprietà dell’oggetto, come terzo la descrizione di tale proprietà (webcomponents, sezione P*olyfills*).

Un oggetto in JavaScript assume differenti stati durante il suo ciclo di vita e nelle fasi di creazione di un Custom element questi vengono definiti tramite quattro metodi: *createdCallback* quando viene creato un Custom element, *attachedCallback* quando viene inserito nel DOM, *detachedCallback* quando viene rimosso dal DOM, e *attributeChangedCallback* quando cambia un attributo. Questefunzioni consentono di manipolare il DOM quando si vuol far accadere un evento sul Custom element.

Una volta creato il prototipo provvisto delle proprietà e dei metodi personalizzati, questo dev’essere assegnato al nuovo elemento per essere registrato nel DOM. la registrazione avviene tramite il metodo *document.registerElement()*, dove i parametri passati saranno necessariamente il nome del nuovo tag (condizione obbligatoria è aggiungere al suo interno il segno “-“) e le opzioni di configurazione. Quest’ultima chiamata avverte il browser del nuovo elemento.

var objectPrototype = Object.create(HTMLElement.prototype);

Object.defineProperty(objectPrototype, ‘title’, {

writable: true

});

objectPrototype.createdCallback = function(){

this.innerText = this.title;

});

var myNameElement = document.registerElement(“my-name”, {

prototype: objectPrototype

});

my-name.html

Viene considerato un Custom element anche l’estensione di un elemento esistente dell’HTML a cui va aggiunto nel tag originale l’attributo *is* (es: <button is=”mega-button”></button>), e registrato all’interno della chiamata *document.registerElement().*

var objectPrototype = Object.create(HTMLElement.prototype),

italicElement = document.registerElement('italic-message’); {

prototype: ObjectPrototype,

extends: ‘i’

});

L’innovazione dei Custom elements risiede nella possibilità di racchiudere funzionalità su misura nell’elemento definendo proprietà e metodi in base alle proprie necessità, come se fossero delle API pubbliche dedicate. Sono loro il cuore dei Web components, perché con la semantica dei tag esprimono tutto il loro potenziale, dalla facilità di costruire nuovi elementi, riutilizzabili e associabili con altri, alla pratica configurazione tramite attributi e funzioni, alla compatibilità con il DOM.

**3.3 SHADOW DOM**

Durante lo sviluppo di un sito o di un’applicazione web, all’aumentare della sua complessità, aumenta anche il rischio di collisione fra le diverse regole e dichiarazioni, specie se si ricorre a librerie esterne per aggiungere funzioni e decorazioni al progetto. Le regole dei fogli di stile o le funzioni JavaScript importate possono contenere selettori, identificatori e regole simili a quelle già presenti nel documento, con il risultato di inaspettate modifiche e sconvolgimenti degli elementi della pagina. Per arginare questo problema, il modello a componenti sfrutta le specifiche w3c dello Shadow DOM. In questo modo è possibile incapsulare il markup, i fogli di stile e le caratteristiche comportamentali di un elemento personalizzato dentro un sotto-albero (una struttura nidificata) indipendente e nascosto rispetto al DOM principale. Questo sistema separa nettamente ogni elemento personalizzato dalle regole del documento ospitante, impedendo di influenzarsi a vicenda: così le dichiarazioni all’interno dello Shadow DOM di un elemento personalizzato non modificano gli altri elementi del file HTML, mentre i file CSS non rielaborano lo stile dell’elemento che possiede un proprio Shadow DOM.

Un file HTML è rappresentato dal DOM con una struttura ad albero dove ogni nodo corrisponde ad un elemento HTML del file. Quando uno di questi nodi coincide con un Custom element, esso risulta il nodo padre o radice del suo sotto-albero (lo Shadow DOM). Il DOM risulta di fatto una foresta di sotto-alberi indipendenti e nascosti che provvedono solo alla resa del loro elemento radice.

Questa funzione sottolinea l’approccio di fondo che il Web component sta portando avanti: separare e rendere autonomi tutti i componenti, renderli riutilizzabili per guadagnare spazio e tempo nella creazione di un’applicazione web o mobile, partire dal contenuto concreto per cambiare la mentalità e il lavoro del programmatore web.

Per creare uno Shadow DOM bisogna selezionare un elemento e chiamare il suo metodo *createShadowRoot*: restituirà un frammento di documento che sarà possibile riempire di contenuto. Nel linguaggio dello Shadow DOM l’elemento che viene richiamato con il metodo è detto *Shadow host*, il solo frammento visibile all’utente, mentre ciò che viene restituito è chiamato *Shadow root,* primo dei nodi discendenti dell’elemento host. Lo Shadowroote i suoi discendenti contengono il contenuto vero e proprio dello Shadow DOM, nascondono all’utente il loro markup ma sono visibili al browser che li restituirà effettivamente sullo schermo una volta elaborati.

<div id=”elem”></div>

<script>

var host = document.querySelector('#elem’);

var root = host.createShadowRoot();

var div = document.createElement(‘div’);

div.textContent = ‘This is Shadow DOM’;

root.appendChild(div);

</script>

Quando invece si vuole aggiungere all’elemento un comportamento più dinamico rendendo possibile l’interazione con input esterni, la tecnologia dello Shadow DOM supporta l’uso dell’elemento chiamato <content>, aggiungendo elementi figli all’albero dello Shadow Host.

var host = document.querySelector('#host');

var root = host.createShadowRoot();

var content = document.createElement('content');

content.setAttribute('select', 'h1'); // <content select="h1"></content>

root.appendChild(content);

*In questo caso, aggiungendo all’elemento content l’attributo select e il selettore h1, è possibile distribuire il contenuto dell’host da un’altra pagina.*

L’incapsulamento e le gerarchie degli alberi dello Shadow DOM rendono ancora più funzionale la separazione tra contenuto e presentazione, snellendo i documenti di un progetto e nascondendo i frammenti di stile. Lo Shadow DOM è progettato per prevenire che qualcuno modifichi accidentalmente l’aspetto dei componenti. Per modificare volontariamente l’aspetto di un elemento dentro ad un componente, bisogna usare delle pseudo-classi come *:host* o *:host-content(),* oppure degli pseudo-elementi come *::content* o *::shadow*, dentro al relativo tag <style>. Con questo selettore si comunica al browser che si vuole attraversare il confine invisibile attorno al componente (HTML5rocks, sezione *Shadow DOM*).

**3.4 TEMPLATE**

L’approccio imperativo dello Shadow DOM per inserire il contenuto all’interno degli elementi non è sempre efficiente e user-friendly. Con i Web components è possibile definire i contenuti dello Shadow DOM in modo dichiarativo con l’HTML inserendoli dentro all’elemento <template>. Questo metodo, usato in passato principalmente lato server per assegnare variabili o per certi comportamenti condizionali, da qualche tempo sta diventando oggetto di lavoro anche lato client tramite i browser, specialmente in una fase in cui stanno mutando gli scenari dell’architettura web: i server si stanno dedicando sempre più all’elaborazione dei dati mentre le interazioni con l’utente ormai avvengo in locale, basti pensare all’utilizzo che ne fanno alcune librerie JavaScript o a certi framework.

I template consentono di memorizzare frammenti di markup, script e fogli di stile inizialmente inerti che vengono analizzati come HTML senza interpretarlo nel caricamento della pagina, per essere attivati all’occorrenza. Anche nei template quindi è preponderante il concetto di creare dei moduli riutilizzabili. L’uso di un template prevede solo l’inserimento del contenuto entro il tag <template>, che non necessita di essere prima registrato perché fa già parte dello standard HTML5, mentre l’attivazione delle sue funzioni avviene in due modi: o con una funzione d’importazione o con una funzione di clonazione:

<script>

var template = document.querySelector('#template');

var container = document.getElementById(“#container”);

var templateContent = template.content;

var clone = document.importNode(templateContent, true);

container.appendChild(clone);

</script>

*Assumendo che #template sia l’id del tag <template> e #container l’id di un <div> generico*

<script>

var template = document.querySelector('#template');

var container = document.getElementById(“#container”);

var templateContent = template.content;

var clone = templateContent.cloneNode(true);

container.appendChild(clone);

</script>

*Assumendo che #template sia l’id del tag <template> e #container l’id di un <div> generico*

Essendo il template un contenitore, una volta interrogato viene inserito il suo contenuto in un nuovo nodo dentro al div che lo visualizzerà sullo schermo.

**3.5 HTML IMPORT**

Gli HTML Import consentono di importare documenti HTML esterni in quello corrente. In questo modo i documenti HTML diventano veri e propri pacchetti di HTML, CSS e JavaScript riutilizzabili e condivisibili. Per usufruire di uno di questi pacchetti è necessario dichiarare nell’intestazione del proprio progetto la seguente dicitura <link rel=”import” href=”x-foo.html”> dove nell’attributo *href* verrà specificato il percorso file corretto in base al suo collocamento.

Infine, combinando insieme le varie tecnologie dei Web components sarà possibile in poche e semplici righe costruire moduli solidi e riutilizzabili, permettendo a chiunque di ampliare da solo le regole dell’HTML o scaricando da altre sorgenti pacchetti già pronti per l’uso previa l’importazione del link nel proprio progetto di lavoro. Quest’approccio alla programmazione web e mobile potrebbe già da ora soppiantare uno stile lineare e verboso per uno modulare e leggero. Esistono già diversi frameworks di supporto per facilitare la creazione di elementi personalizzati, come Polymer, AngularJS, Ember.js, ma anche comunità di programmatori che offrono una vasta libreria di moduli da scaricare come customelements.io o Component Kitrchen.

Con i Web components abbiamo un ricco e disponibile ecosistema a disposizione, tutto incapsulabile in un tag personalizzato.

Esempio completo:

...

<link rel="import" href="x-component.html">

</head>

<body>

<x-component>

Mondo!

</x-component>

...

index.html

<template id="template">

<style>

...

</style>

<div id="container">

<p> Ciao <content></content>. Questo è un elemento dentro lo Shadow DOM dell’elemento personalizzato</p>

</div>

</template>

<script>

var XComponent = document.registerElement('x-component', {

prototype: Object.create(HTMLElement.prototype, {

createdCallback: {

value: function() {

var root = this.createShadowRoot();

var template = document.querySelector('#template');

var clone = document.importNode(template.content, true);

root.appendChild(clone);

}

}

})

});

</script>

x-component.html

**4. Polymer**

**4.1 Architettura di Polymer**

Polymer definisce un nuovo approccio alla progettazione e programmazione web. È una libreria che consente l’utilizzo e l’implementazione delle specifiche dei Web component con lo scopo di semplificare e velocizzare la creazione di elementi integrabili e riutilizzabili per il web. L’intenzione è di applicare al proprio lavoro un’impostazione modulare dove piccoli tasselli formano insieme un complesso più grande e riutilizzabile. Polymer si basa sulle norme del Web component e aiuta l’utente nella creazione di elementi personalizzati oppure nell’utilizzo di componenti già presenti nel catalogo Polymer, fornendo una sintassi dichiarativa per rendere più semplice la loro dichiarazione.

L’architettura di questa libreria si fonda su tre differenti livelli strutturali:

* Foundation: una collezione di polyfills (librerie JavaScript) che consentono ai browser che non ne dispongono, di supportare le specifiche del Web component indispensabili per la loro lettura;
* Core: l’insieme delle informazioni e delle funzionalità che permettono di sfruttare l’intera tecnologia; fornisce una sintassi dichiarativa per costruire i propri elementi personalizzati;
* Elements: libreria dei componenti esistenti da importare o da creare e condividere.

Polymer si serve del DOM dei browser come di un framework: dal momento che il *Document Object Model* ha insite alcune caratteristiche ad esso attinenti, anziché riprogettarle e riprodurle nuovamente su una piattaforma web, i programmatori di Polymer hanno guardato a queste parti utili del DOM per usarle a beneficio del programmatore che non dovrà imparare ulteriori regole. Una delle caratteristiche sfruttabili del DOM è di essere già dotato di un proprio modello a componenti, dove questi componenti sono gli elementi (basti pensare all’elemento <video> che, seppur possieda un’interfaccia molto semplice e facile da usare, nasconde una grossa complessità al suo interno). L’interazione col DOM avviene tramite i Custom element. La separazione dei concetti è la chiave di un’applicazione scalabile, che a partire da un elemento personalizzato unito ad altri elementi forma una pagina, e insieme ad altre pagine creano un’applicazione: tutto assemblabile, tutto riutilizzabile e facilmente mantenibile. Una delle filosofie conclamate di Polymer è “*don’t repeat yourself*” e ridurre la scrittura di codice già esistente, dando vita a pagine ordinate e sintetiche (Polymer-project.org, sezione *Guides & resources*).

**4.2 Elementi personalizzati**

**4.2.1 Uso della libreria**

Due sono le modalità di lavoro con Polymer: usare gli elementi della libreria oppure crearne di nuovi. Nel primo caso è previsto nella sezione *Browse Element Catalog* una vasta scelta di elementi divisi per categorie, tra cui una sezione dedicata all’estensione per il Material design. Qui è possibile trovare tutti i componenti proposti dal team di Google con le relative proprietà e funzioni. Per utilizzarle è necessario scaricare nel proprio progetto la cartella contente la collezione di APIs ed importare tramite la procedura prevista dai Web component l’elemento desiderato. Questo sistema è il modo più veloce e sicuro per poter usufruire di un elemento testato e adattabile ad ogni dispositivo senza che le sue specifiche influenzino quelle del documento principale. Oltre all’estensione del Material design (definito nell’elenco come *Paper Element*) è possibile attingere fra le seguenti categorie: *Iron Elements,* un insieme di elementi visivi e non visivi per lavorare con i layout, componenti standard, chiamate AJAX; i *Google Web Components,* ovvero pacchetti che sfruttano alcune funzioni delle applicazioni di Google come Maps, Chart, Analytic; i *Gold Elements* per costruire applicazioni e-commerce; *Neon Elements* per animazioni di base; *Platinum Elements* per estendere alcune funzionalità come la navigazione off-line o le tecnologie bluetooth; *Molecules*, elementi per interagire con altre librerie JavaScript.

**4.2.2 Creare un elemento**

Nel secondo approccio Polymer estende e semplifica la creazione dei Custom elements: in questo caso, rispetto allo standard proposto dai Web component, il procedimento risulta più immediato e intuitivo, forte dell’approccio dichiarativo e ad alto livello proposto. Per registrare un nuovo elemento (il cui nome deve contenere il trattino per differenziarlo dagli elementi HTML originali) viene utilizzata la funzione riservata *Polymer*, a cui associare la proprietà *is* con il nome del tag corrispondente. In automatico viene registrato nel browser il nuovo elemento.

Nella funzione vengono inoltre configurate le proprietà dell’elemento tramite l’oggetto *properties*, dal momento che ogni proprietà rientrante in un’API pubblica di un nuovo elemento deve essere dichiarata. Le proprietà vengono supportate dalle seguenti chiavi:

* *Type*, definisce il tipo di contenuto di un dato, estendendo il valore degli attributi di un tag HTML fino ad ora solo stringa;
* *Value,* indica il valore di default assegnato alla proprietà;
* *ReflectToAttribute*, specifica la modifica della proprietà all’attributo;
* ReadOnly, indica quando la proprietà è di sola lettura;
* *Notify*, permette di abilitare il *two-data binding;*
* *Computed*, indica il metodo che viene richiamato per calcolare il valore della proprietà, ogni volta che cambia un argomento;
* *Observer*, indica il metodo da richiamare quando cambia il valore della proprietà.

Per inserire del contenuto il passo successivo è la definizione dichiarativa del DOM locale del componente tramite l’elemento <dom-module>, che corrisponde al principio dello Shadow DOM per il modello dei Web component. Ad esso viene aggiunto l’attributo *id* settato con lo stesso nome che durante la registrazione viene assegnato alla proprietà *is,* per mantenere automaticamente collegate le due definizioni. Ereditato dai Web component, l’elemento <templete> al suo interno racchiude le informazioni eventualmente elaborate attraverso le regole dichiarate nello <style>. Tale meccanismo permette di definire lo stile dell’elemento senza che questo interferisca con il resto del documento dal momento che sfrutta le specifiche dello Shadow DOM.

<link rel=”import” href=”bower\_components/polymer/polymer.html”>

<dom-module id= ‘nuovo-elemento’ >

<template> // contiene lo Shadow DOM dell’elemento

<style>

p {

color: red;

}

</style>

<p>ecco un nuovo elemeto</p>

</template>

</dom-module>

<script>

Polymer ({ //dichiarazione dell’elemento

is : ‘nuovo-elemento’ ,

properties : {

name : {

type : String ,

readOnly : true

}

}

})

</script>

In alternativa, è possibile estendere un elemento HTML nativo: per farlo, è necessario settare la proprietà *extends* con il nome del tag da estendere all’interno del prototype durante la dichiarazione. Per usarlo nel file HTML basta aggiungere l’attributo is all’interno del tag desiderato ( <input is=”new-input”> ).

Un’ulteriore estensione per favorire il riutilizzo degli elementi avviene sui fogli di stile grazie ai *Custom CSS properties*, definendo delle variabili che agiscono sulle regole dello stile di una proprietà o tramite i *Custom CSS mixins* su interi blocchi semplificando il sistema verboso e complesso di pseudo classi di modifica presenti nel Web component. Queste proprietà sovrascrivono il valore di default presenti nel DOM locale di un componente consentendo di modificare anche elementi pensati da altri per adattarsi a ogni progetto. Inoltre, all’occorrenza è possibile importare un intero <dom-module> con lo stesso meccanismo di un normale elemento.

**4.3 Data-binding e <content>**

Il *data binding* (associazione di dati) è il processo di connessione dinamico tra gli oggetti e gli elementi del DOM che anche Polymer ha sposato per arricchire le sue funzioni: questo potente strumento consente di legare le proprietà di un elemento personalizzato con le proprietà o gli attributi degli elementi de DOM locale. Quest’associazione viene dichiarata negli attributi HTML con il nome della proprietà o sotto proprietà da inserire nelle doppie parentesi graffe o quadre a seconda del tipo di associazione (flusso di dati bidirezionale per il primo e unidirezionale per il secondo).

<link rel=”import” href=”bower\_components/polymer/polymer.html”>

<dom-module id= ‘demo-binding’ >

<template>

<div> {{foo}} </div>

<paper-input value=”{{foo}}”></paper-input>

</template>

</dom-module>

<script>

Polymer ({

is : ‘demo-binding’ ,

})

</script>

Nell’esempio riportato Polymer analizza il template e cerca il nodo div con l’annotazione indicato dalle parentesi graffe; una volta trovato crea un oggetto che sa riconoscere quale nodo del DOM aggiornare ogni volta che la proprietà a cui è associato cambia. Essendoci le parentesi graffe, in questo caso, ogni volta che viene digitata una stringa nell’input, cambierà automaticamente sia il contenuto del div che quello nell’input.

Per far interagire il DOM locale con elementi di markup esterni, Polymer supporta l’elemento <content> grazie al quale viene indicato nell’albero del DOM locale il punto in cui inserire altri elementi figli eventualmente specificati nel file index del nuovo progetto. <content> è provvisto dell’attributo *select* per filtrare quale elemento deve sostituirlo (Polymer polytechnic, sezione *resources*).

<link rel=”import” href=”bower\_components/polymer/polymer.html”>

<dom-module id= ‘nuovo-elemento’ >

<template>

<content select=”.content”></content>

</template>

</dom-module>

<script>

Polymer ({

is : ‘nuovo-elemento’ ,

})

</script>

nuovo-elemento.html

<html>

<nuovo-elemento>

<div class=”.content”>contenuto</div>

</nuovo-elemento>

</html>

index.html

<html>

<div class=”.content”>contenuto</div>

<html>

output

Polymer quindi fornisce una serie di funzionalità per semplificare la creazione degli elementi personalizzati o per utilizzare componenti già esistenti all’interno del suo catalogo in costante aggiornamento. Ovviamente le caratteristiche elencate e discusse fino a qui non esauriscono l’intero scenario di Polymer assai più complesso, ma esprimono un primo indispensabile approccio alla materia.

Lo scopo di Polymer è di sbloccare le capacità e le qualità dei Web component proponendo un modo diverso di pensare e sviluppare il web. La funzionalità dell’incapsulamento, il concetto di riutilizzo e condivisione, la versatilità di un approccio dichiarativo alla creazione di elementi sono il cuore e l’anima del web componibile.

1. **Conclusioni**

Realizzare un sito o un’applicazione web è un processo che richiede tempo e campi d’azione differenti: serve avere dei contenuti validi e saperli strutturare con qualità editoriali, riuscire a presentarli graficamente per una comunicazione efficace, combinando questi due aspetti con linguaggi di programmazione adeguati. Tale lavoro, in ambito professionale, viene spesso suddiviso e affidato a più membri specializzati di un équipe, proprio perché ogni aspetto va curato con attenzione e necessita un tempo di elaborazione considerevole. Per sopperire alla mancanza di un gruppo di lavoro risulta utile affidarsi a strumenti e modelli consolidati con l’intento di alleggerire e velocizzare il carico senza perdita di qualità. L’intento di questo progetto di laurea è far conoscere ad un aspirante sviluppatore web alcune linee guida recentemente studiate sia per migliorare la progettazione e la programmazione web sia l’esperienza utente che ne deriva. Applicare i principi di usabilità e di resa grafica proposti dal Material design consente di valorizzare la comprensione dei contenuti di un sito e di avviare un processo di standardizzazione di buone abitudini atte a migliorare la navigazione web. Progettare con il modello a componenti (adiuvato da solide librerie) rende veloce e ordinata la scrittura di codice, lasciando il tempo al progettatore di concentrare l’attenzione sugli obiettivi e problematiche strettamente legate al proprio sito.

I limiti di un tutorial possono essere ascrivibili entro due aspetti: offrire poca possibilità di mettere in pratica la teoria esposta e non coprire in modo esaustivo l’intero argomento; assume qui il valore di introduzione ai temi trattati per stimolare alla consultazione delle rispettive documentazioni in rete. Modificare abitudini di programmazione e utilizzare nuove tecnologie, come ogni cambiamento, necessita un periodo adeguato per comprenderne il funzionamento e assimilarne i meccanismi, non senza una buona dose di esercizio ed esperienza sul campo. Infine, essendo tecnologie emergenti, è probabile che entro un breve periodo esse possano ricevere modifiche e aggiornamenti: è quindi auspicabile un lavoro di manutenzione del tutorial per adeguare le informazioni alle nuove versioni.

# **Riferimenti**

* 1. **Bibliografia**

Giannini, Anna Maria. Tessa Marzi, Maria Pia Viggiano. *Design. Percezione visiva e cognizione, psicologia dell'arte, la scelta del prodotto: emozioni, decisioni e neuroestetica.* Giunti editore spa. 2011

Krug, Steve. *Don't make me think, revisited a common sense approach to web usability.* New Riders. 2014

Liscia, Roberto. (a cura di). *E-learning. Stato dell'arte e prospettive di sviluppo.* Milano: Apogeo srl. 2004. pp. 12-18

Patel, Sandeep Kumar. *Learning Web component development.* Packt publishing. 2015

**6.2 Sitografia**

Component.kitchen.

http://component.kitchen/tutorial [consultato il 13 ottobre 2015]

CSS tricks.

https://css-tricks.com/modular-future-web-components/ [consultato il 7 ottobre 2015]

F2.net.

http://www.f2.net/ux-design-principi-fondamentali-introduzione/ [consultato il 27 ottobre 2015]

Froont. Open design blog.

http://blog.froont.com/brief-history-of-web-design-for-designers/ [consultato il 3 settembre 2015]

HTML5rocks.

http://www.html5rocks.com/en/tutorials/?page=1 [consultato il 20 ottobre 2015]

Material design.

https://www.google.com/design/spec/material-design/introduction.html [consultato il 1 novembre 2015]

MDB. Material design blog.

http://materialdesignblog.com/the-ultimate-guide-to-googles-material-design/ [consultato il 10 settembre 2015]

Polymer-project.org

https://www.polymer-project.org/1.0/docs/devguide/feature- overview.html [consultato il 31 ottobre 2015]

Polymer polytechnic.

http://itshackademic.com/ [consultato il 24 ottobre 2015]

TLC. Web solutions. Think. Listen. Change.

http://www.tlcws.com/blog/flat-design-e-scheumorfismo-i-vantaggi-e-gli-svantaggi.html [consultato il 8 settembre 2015]

W3C.

http://www.w3.org/TR/components-intro/ [consultato il 13 ottobre 2015]

Webcomponents.

http://webcomponents.org/polyfills/ [consultato il 21 ottobre 2015]

Duarte, M.

https://www.google.com/events/io/io14videos/79edef8b-96d4-e311-b297-00155d5066d7 (2014, Giugno 25).

1. Formato per immagini digitali e animate [↑](#footnote-ref-1)
2. Scaricabile al seguente link: https://code.google.com/p/io-2012-slides/ [↑](#footnote-ref-2)
3. Per una visione d’insieme, consultare il seguente link: https://www.google.com/design/spec/style/color.html [↑](#footnote-ref-3)
4. Team leader del progetto “Android Design” di Google [↑](#footnote-ref-4)
5. Visibile al seguente link: https://fronteers.nl/congres/2011 [↑](#footnote-ref-5)