ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seconda Facoltà di Ingegneria

Corso di Laurea in INGEGNERIA INFORMATICA

Tesi di Laurea in SISTEMI DISTRIBUITI

Evoluzione dei modelli e delle tecnologie per il web: DART e HTML5 come caso di studio

Candidato:
Laddaga Michele

Relatore:
Prof. Alessandro Ricci

Anno Accademico 2011/2012 - Sessione II

Alle persone che mi hanno aiutato
e söpportato in questo percorso.

Indice

Introduzione

Nell'ultimo decenno si è assisitito ad una frenetica evoluzione delle tecnologie di connessione e trasferimento dati, dagli impianti di fibra ottica sempre piú diffusi e performanti, alle varie tecnologia mobile UMTS e LTE. Tutto ciò ha permesso a sempre piú persone di poter spendere una maggiore fetta del proprio tempo sulla rete sia per svago che per lavoro.
Questo ha portato col tempo l'utente interattivo alla ricerca di servizi remoti che prima usufruiva in locale, sia per motivi di portabilità e di interconnessione, sia per una semplice questione di sincronizzazione fra i vari dispoditivi posseduti.

Il presente progetto di tesi si pone come obiettivo di indagare con occhio ingegneristico all'evoluzione della struttura del web fino ad identificare e analizzare l'attuale necessità di poter avere in rete tutti quei servizi anche completi e strutturalmente complessi che prima si aveva sul proprio desktop; tutto ciò attraverso l'esempio di un nuovo linguaggio di sviluppo per applicazioni web strutturate proposto da Google: DART.
In questa analisi non si potrà prescindere dallo studio attento della tecnologia che sin dagli inizi della rete ha fatto da struttura al web ovvero l'Html e il suo ultimo standard Hmtl5.

Nella prima parte verrà mostrata, attraverso un breve percorso, la nascita e lo sviluppo del web, sino ai giorni nostri.
Si effettuerà quindi una panoramica, rivisitando tutti più importanti passi che hanno portato la rete internet ad essere ciò che oggi utilizziamo, ponendo una particolare attenzione alle attuali esigenze di progettazione della struttura dei servizi web.

Nel secondo capitolo viene introdotta la tecnologia a base del web, l'Html; attraverso una breve analisi dell'evoluzione di quest'ultima si arriverà sino all'attuale HTML5 e alle funzionalità offerte nell'ottica della programmazione web strutturata.
Nel terzo capitolo si analizzerà in maniera più approfondita la necessità di un web strutturato e le potenzialità del progetto Dart attraverso alcuni esempi esplicativi.

Infine si trarranno alcune conclusioni sull'attuale situazione del web, sulla necessità di un web strutturato e sulla possibilità di riuscita del progetto Dart.

Capitolo 1

Evoluzione del WEB

1 La nascita di Internet

Il quarto giorno di Ottobre del 1957 l'Unione Sovietica lanciò con successo il primo satellite nell'orbita terrestre. Questo evento porterà direttamente alla creazione del Dipartimento della Difesa ARPA (Advanced Research Projects Agency), a causa di una riconosciuta necessità di una organizzazione per la ricerca e lo sviluppo di idee e tecnologie avanzate, indipendentemente dai bisogni del momento. Forse il loro progetto più famoso (certamente il più grande mai utilizzato) è stato la creazione di Internet.
Il progetto per il network (definito Arpanet), fu presentato nel mese di ottobre 1967, e nel dicembre 1969 i primi quattro computer in rete furono installati e subito furono perfettamente funzionanti. Nel 1982 le connessioni ARPANET al di fuori degli USA furono convertite e fu utilizzato il nuovo protocollo TCP/IP (Transmission Control Protocol / Internet Protocol). Internet, come lo conosciamo oggi, era arrivato.

1.1 L'alba del www

Gopher è stato un sistema di recupero delle informazioni utilizzate nei primi anni '90, implementando una serie di menu collegati a file, risorse del computer e altri menu. Questi collegamenti potevano oltrepassare i confini dei computer del tempo e utilizzare Internet per recuperare informazioni da altri sistemi. E' stato un successo nelle università come mezzo per

condividere informazioni di grandi dimensioni al fine di rendere disponibile un documento per l'archiviazione e la gestione.
Gopher è stato creato dalla Università del Minnesota. Nel febbraio del 1993 fu annunciato che avrebbero tassato la concessione dell'uso dei server di Gopher. Di conseguenza, molte organizzazioni cominciarono a pensare a sistemi alternativi a Gopher stesso.
Il Consiglio Europeo per la Ricerca Nucleare (CERN) - Svizzera - ha pensato ad una alternativa. Tim Berners-Lee lavorò ad un sistema di gestione delle informazioni, in cui il testo avrebbe potuto contenere link e riferimenti ad altri testi, permettendo al lettore di poter passare di documento in documento. Aveva creato un server per poter pubblicare questo modello di documento (definito ipertesto), nonché un programma per la lettura dei testi: il WorldWideWeb. Questo software fu rilasciato la prima volta nel 1991.
Nel mese di Aprile del 1993 il CERN rese di pubblico dominio il codice sorgente del WorldWideWeb, in modo tale che chiunque avrebbe potuto usare o sviluppare il software senza nessuna somma da pagare.

1.2 La guerra dei Browser : serve un nuovo standard

La diffusione del Web ebbe risvolti, ed interessi, commerciali. Marc Andreessen lasciò la NCSA ed assieme a Jim Clark fondò la Mosaic Communications, successivamente rinominata Netscape Communications Corporation, la quale iniziò a lavorare su quello che sarebbe divenuto Netscape Navigator. La versione 1.0 del software fu rilasciata nel dicembre 1994. Spyglass Inc (il braccio commerciale della NCSA) brevettò la tecnologia di Mosaic con la Microsoft per costituire le basi di Internet Explorer. La versione 1.0 fu rilasciata nel mese di agosto 1995. Seguì presto una rapida escalation affinchè Netscape e Microsoft prevalessero l'uno sull'altro e per attirare gli sviluppatori. Tutto questo ha preso il nome della guerra dei browser. Durante la guerra dei browser piuttosto che correggere i problemi delle features supportate, Netscape e Microsoft si concentrarono sulla realizzazione di nuove caratteristiche, aggiungendo funzionalità proprietarie e creando funzioni che sono state in concorrenza diretta con

caratteristiche esistenti negli altri browser, ma realizzati in maniera incompatibile. Gli sviluppatori sono stati costretti a far fronte a crescenti livelli di confusione quando si è trattato di costruire siti web costruendo, a volte, due copie dello stesso sito per i due maggiori browser; in alcuni casi supportando le caratteristiche di un solo browser, in altri ”bloccando“ a terzi la visualizzazione dei loro siti web. Questo è stato un terribile modo di lavorare, con l'inevitabile contraccolpo da parte degli sviluppatori.

Nasce il w3c

Nel 1994 Tim Berners-Lee fondò il World Wide Web Consortium (W3C) presso il Massachusetts Institute of Technology, con il supporto del CERN, DARPA (inizialmente ARPA) e la Commissione europea. L'intento del W3C è stata la standardizzazione dei protocolli e delle tecnologie utilizzate per costruire il web in modo tale che i contenuti fossero a disposizione di più gente possibile.
Dalla sua nascita, il W3C ha pubblicato numerose specifiche (chiamate raccomandazioni) tra cui le varie versione dell'HTML , il formato per immagini PNG ed i Cascading Style Sheets (CSS) versioni 1 e 2. Esso ha lo scopo di guidare lo sviluppo del Web e di definirne gli standard, dunque di identificare i requisiti tecnici perché il Web diventi un ”universal information space“.
Gli obiettivi e i principi strategici della W3C possono essere spiegati mediante 7 punti fondamentali:

- Accesso universale;
- Web semantico;
- Fiducia;
- Interoperabilità;
- Capacità evolutiva;
- Decentralizzazione;
- Multimedia coinvolgente .

Tuttavia quelle del W3C sono solo linee guida; gli sviluppatori devono creare prodotti conformi a tali raccomandazioni e, se lo desiderano, apporre

immagini di ”etichettatura“ che attestano la conformità. In pratica, questo non è un buon obiettivo se non è portato a conoscenza della maggior parte degli utenti.
Nel 1998 il mercato dei browser è stato dominato da Internet Explorer 4 e Netscape Navigator 4. Viene rilasciata una versione beta di Internet Explorer 5 il quale implementa un nuovo e proprietario dynamic HTML. Ciò significa che gli sviluppatori web necessitano di conoscere ben 5 modi diversi di scrivere il JavaScript. Qualcosa va cambiato.

2 Evoluzione del Web: necessità di applicazioni Web Strutturate

Il Web 2.0 è la rete come piattaforma.
Attraverso tutti i dispositivi collegati,
le applicazioni Web 2.0 permettono
di ottenere la maggior parte dei vantaggi intrinseci della piattaforma,
fornendo il software come un servizio in continuo aggiornamento,
che migliora più le persone lo utilizzano,
sfruttando diverse combinazioni di dati
provenienti da sorgenti multiple. - Tim O'Reilly -

Il Web degli albori era formato da un insieme di pagine statiche alla quale ci si collegava per trarre informazioni testuali e immagini senza la possibilità di interazione con esso. Erano documenti HTML dove all'interno si trovavano le informazioni da visualizzare. Si trattava quindi di un web passivo dove una volta raccolta l'informazione, l'interazione terminava. Successivamente in breve tempo si è verificata un'evoluzione del Web molto significativa: dall'iniziale WEB 1.0 si passò, attraverso l'integrazione dei data base, alla 1.5 con la nascita dei primi forum e blog anche se ancora acerbi, sino ad arrivare al web dei giorni nostri denominato WEB 2.0; quest'ultimo rappresenta l'insieme di tutte quelle applicazioni online che permettono uno spiccato livello di interazione sito-utente (blog, forum, chat, sistemi quali Wikipedia, Youtube, Facebook, Myspace, Twitter, Gmail, Wordpress,

Tripadvisor ecc). Grazie a tale evoluzione anche la creazione di contenuti è diventata alla portata di tutti; oggi non c'è più bisogno di conoscenze approfondite sulle tecnologie web per avere un blog o condividere filmati e contenuti digitali in genere.

2.1 Il Web Interattivo

I siti del Web 2.0 permettono agli utenti un grado d'interattività assoluto, trasformando il ruolo degli utenti, da semplici visitatori del sito a veri e propri protagonisti del Web; si vedano come esempi lampanti le varie piattaforme di condivisione come youtube o wordpress o, più esplicativo di tutti, il progetto Wiki che basa la propria fortuna sulla partecipazione attiva di moltissimi utenti della rete per creare quella che ad oggi risulta l'enciclopedia digitale più completa al mondo. Queste interazioni però possono aver luogo grazie alla creazione di siti e servizi sempre più complessi da parte dei programmatori del web. Attraverso varie tecnologie quali ad esempio Ajax che utlizza il javascript e Flash è stato possibile creare questa interazione con l'utente: chi naviga può inviare e ricevere informazioni senza che la pagina venga ricaricata. Per quanto riguarda la gestione dei servizi dal lato server, il Web 2.0 utilizza tecnologie come PHP, Ruby, Perl, Python, JSP, permettendo agli sviluppatori di manipolare dinamicamente informazioni provenienti da file e database. Molto importante è la tecnica di formattazione dei dati per lo scambio di questi nella rete ovvero l'XML e JSON. Importanti nel Web 2.0 sono gli "slates" ovvero determinate caratteristiche e tecniche qui riproposte:

- **Search** - recuperare informazioni tramite una parola chiave;
- **Links** - collegare le informazion;
- **Authoring** - creare e aggiornare i contenuti porta alla partecipazione di innumerevoli autori;
- **Tags** - parola chiave per facilitare la ricerca;
- **Extentions** - software che rende il Web una piattaforma applica-tiva invece che un semplice server di documenti;
- **Signals** - utilizzo di tecnologie come RSS per noti care all'utente cambiamenti nel contenuto;

Utilizzando tali metodi si sta andando verso un web sempre più completo e complesso che necessità quindi di un diverso approccio per la creazione dei servizi.

2.2 Applicazioni Web Strutturate

Grazie al sempre maggiore numero di persone connesse, il Web sta velocemente invadendo ogni campo dall'informazione, al sociale, all'intrattenimento, al business. Con l'aumentare del tempo medio trascorso sulla rete l'utente necessita di poter avere sempre con se tuttii quei servizi che prima utilizzava in locale: i programmatori hanno iniziato a trasportare sulla rete tutte le applicazioni che un tempo erano utilizzate sul proprio computer, in modo da poter dare ogni tipologia di servizio in qualunque luogo ci si trovi, ed in maniera sincronizzata tra i vari dispositivi, a patto di potersi collegare al WWW. Fa parte di ciò il cloud computing ovvero l'insieme delle tecnologie che permettono, tipicamente sotto forma di un servizio offerto da un provider al cliente, di memorizzare/archiviare e/o elaborare dati (tramite CPU o software) grazie all'utilizzo di risorse hardware/software distribuite e virtualizzate in Rete. Un'applicazione desktop è fisicamente eseguita sul computer e utilizza le risorse della macchina su cui è installata e per le applicazioni che hanno un considerevole carico computazionale, sono richiesti requisiti hardware ben definiti. Al contrario, poiché un'applicazione cloud è resa fruibile attraverso la connessione a un altro sistema, il carico computazionale è quasi interamente gestito dal service provider che offre il servizio, riducendo i requisiti hardware della macchina che lo utilizza. Non sarà più necessario quindi un hardware particolarmente potente perchè migreranno online tutte quelle attività che fino a poco tempo fa erano necessariamente svolte in locale. I dati che erano salvati sui computer saranno decentrati su vari server e accessibili grazie al browser garantendo diversi vantaggi :

- una diminuzione dei costi per l'acquisto del computer, non essendoci più bisogno di particolare potenza computazionale;
- possibilità di accedere al sistema in qualsiasi momento, anche in viaggio, grazie ad una connessione mobile.

- i dati sono completamente gestiti dal service provider che ne garantisce il corretto backup, questo evita la perdita di dati legata a un possibile guasto del nostro hard diski;

Il cloud computing tuttavia comporta anche alcuni aspetti negativi :

- necessità di una connessione veloce non ancora disponibile ovunque;
- è obbligatorio l'utilizzo di tecniche di protezione dei dati come la cifratura per garantire la sicurezza di questi;
- tutto si basa sul presupposto che il servizio resti sempre disponibile e online;

Esempi importanti di questa tecnologia si hanno dalle maggiori società che operano nell'informatica:

- **Microsoft** offre una serie di servizi software online gratuiti rendendo possibile l'accesso via web a numerose funzionalita che si arricchiscono ogni giorno; l'idea finale è quella di trasferire direttamente il sistema operativo sulla rete attraverso servizi e modelli(ne è un primo passo il nuovo Windows 8);
- **Apple** col suo iCloud, permette di accedere ai propri dati direttamente dallo smartphone, tablet,portatile o fisso, consentendo di sincronizzare contatti, documenti, email, applicazioni, impostazioni, immagini e altro in modo del tutto automatico.
- **Google** attraverso la GMail e tutti i servizi Google Apps che come per ICloud si possono sincronizzare con i vari dispositivi a disposizione.

Tutte questi servizi però necessitano di **un nuovo metodo di creazione del software metodico e strutturato** come già avviene per le grosse applicazioni create sino ad oggi in locale. La mancanza di una struttura negli script, realizzati all'interno di applicazioni web di grandi dimensioni, portano a problemi di correzione e mantenimento del codice non che di creazione dello stesso. Inoltre queste applicazioni non possono essere suddivise in modo tale da assegnare i vari compiti all'interno di una squadra di lavoro. Attualmente quando si crea un servizio si parte con piccoli script che

rapidamente evolvono in grandi applicazioni senza una struttura ben definita, cosa che rende difficile il debug e la manutenzione dell'applicazione stessa come detto poc'anzi.

Capitolo 2

HTML5

HTML è l'acronimo di Hypertext Markup Language ("Linguaggio di contrassegno per gli Ipertesti"); non si tratta di un linguaggio di programmazione (in quanto non prevede alcuna definizione di variabili, strutture dati, funzioni, strutture di controllo) ma solamente un linguaggio di markup che descrive le modalità di impaginazione, formattazione o visualizzazione grafica (layout) del contenuto, testuale e non, di una pagina web attraverso tag di formattazione.
Tuttavia, l'HTML supporta l'inserimento di script e oggetti esterni quali immagini o filmati come in seguito si vedrà.

1 Evoluzione dell'HTML

Quando il Cern ebbe la necessità di comunicare con le vari sedi, si potevano già utilizzare servizi Internet quali FTP e posta elettronica, ma per rendere lo scambio di informazioni semplice e intuitiva a chiunque non avesse familiarità con le reti di calcolatori, progettarono e costruirono un sistema di condivisione dei documenti usando un concetto chiamato ipertesto (anche se la prima idea di sistema ipertestuale si deve far risalire a Vannevar Bush nel 1945). L'ipertesto era costituito da vari documenti connessi tramite link. Questo permise ai ricercatori di condividere le documentazioni scientifiche indipendentemente dalla piattaforma usata; infat-

ti Tim Berners-Lee e Robert Cailliau definirono standard e protocolli per scambiare documenti su rete, progettando il linguaggio HTML e il protocollo http.
Inizialmente gli standard e protocolli supportavano solo la gestione di pagine HTML statiche, cioè navigabili con opportune applicazioni browser, ma subito dopo, furono progettati strumenti capaci di costruire pagine HTML dinamicamente, aumentando le potenzialità e le funzionalità dei browser grazie ad un'evoluzione del linguaggio. Così, negli ultimi anni l'HTML ha subito molte revisioni e miglioramenti, passando dalla versione 1.0 , alla 2.0, alla 3.2 e arrivando alla versione 4.0 e 4.01. Per esempio la versione HTML 3.2, utilizzata dai cosiddetti browser di terza generazione, permetteva di regolare gli allineamenti delle celle della tabella al punto, ed è l'unico contesto in cui gli allineamenti si assegnavano per pixel; permetteva inoltre di allargare una cella per occupare le colonne limitrofe o le righe limitrofe, migliorando così, rispetto alla precedente versione il lavoro dei designer. Dalla versione 4.0 si può separare contenitore da contenuto, aggiungere supporto per nuove tecnologie(CSS, JavaScript e jQuery, XML, JSON,..), migliorare l'accesso web ai portatori di handicap e molto altro.

1.1 L'XHTML

Il 26 gennaio 2000 il W3C rilascia la prima specifica del linguaggio di markup destinato a sostituire HTML. Quel linguaggio si chiama XHTML. L'XHTML (acronimo di eXtensible HyperText Markup Language, Linguaggio di marcatura di ipertesti estensibile) è un linguaggio di marcatura che associa alcune proprietà dell'XML(Extensible Markup Language) con le caratteristiche dell'HTML: un file XHTML è un pagina HTML scritta in conformità con lo standard XML. Il linguaggio prevede un uso più restrittivo dei tag HTML sia in termini di validità che in termini di sintassi per descrivere solo la struttura logica della pagina, mentre il layout e la resa grafica sono imposti dai fogli di stile a cascata (Cascading Style Sheets, CSS). Tecnicamente può essere definito come una riformulazione dell'HTML 4.01 in XML 1.0 e quindi una sorta di ponte tra questi due linguaggi. L'elemento chiave è la X. Sta per EXTENSIBLE, è la stessa X su cui si fonda quella che è

probabilmente la pietra angolare della comunicazione digitale del futuro: XML. La cosa ”rivoluzionaria“ è appunto questa: HTML è ora una parte della grande famiglia XML, ne condivide regole di base e potenzialità.

I Vantaggi

Codice pulito e ben strutturato

HTML è nato come linguaggio per definire la struttura di un documento. Non ha nulla a che vedere con la presentazione. Eppure, è stato modificato nel tempo per svolgere compiti per cui non era stato ideato. Il caso delle tabelle è quello più eclatante: esse sono nate per la presentazione di dati tabulari ma sono state impiegate come l'unico mezzo per costruire il layout della pagina. Un altro esempio si trova nell'introduzione del tag ¡font¿ per gestire la tipografia dei documenti; la conseguenza fu quella di ritrovarsi con pagine cariche di elementi inutili, pesanti e difficili da modificare. La responsabilità principale per questo uso improprio di HTML non ricade sugli sviluppatori. Nel 1996 il W3C ha rilasciato la versione definitiva di CSS1. Era questa la tecnologia destinata a definire la presentazione dei documenti con la chiara idea di avere l'HTML per la struttura e il CSS per lo stile e il layout. Eppure, per avere browser che supportano in maniera accettabile questi standard si è dovuto attendere il 2000-2001. Quattro anni, tre generazioni di browser, milioni di siti costruiti tentando di risolvere incompatibilità e bug. Con XHTML, almeno nella sua versione Strict, si torna ad un linguaggio che definisce solo la struttura. Semplicemente, se vi si inseriscono elementi non supportati (font, larghezza per le celle di tabelle o margini per il body, per citare solo alcuni esempi) il documento non è valido. Quindi: la formattazione si fa con i CSS col risultato di un codice più pulito, più logico e più gestibile.

Portabilità

Portabilità è la capacità/possibilità di un documento di essere visualizzato e implementato efficacemente su diversi sistemi: PC, PDA, cellulari WAP/GPRS, WebTV. Ora, se si pensa alle limitazioni in termini di memoria, ampiezza dello schermo e usabilità di un terminale mobile, si capisce l'importante un linguaggio essenziale, centrato sulla struttura del documento;

qualcosa con un titolo della pagina, un'intestazione, un paragrafo e una lista per scandire gli argomenti. Sulla portabilità poggia l'enfasi con cui aziende del calibro di Nokia, Motorola, Ericsson o Siemens guardano ad XHTML. Dopo l'accettazione da parte del Wap Forum si può affermare con certezza che la piattaforma WAP 2 e tutta l'evoluzione dei servizi mobili sarà fondata sull'integrazione tra XHTML e CSS, con il supporto delle necessarie tecnologie sul lato server. L'intento di alcune società come Nokia è molto chiaro :

- nella pagina XHTML si incorporano diversi CSS per ciascun supporto
- il browser viene identificato
- su un PC si vedrà il layout standard
- su un cellulare si visualizzerà un layout ridotto e adatto alle caratteristiche del mezzo
- ciò che non cambia sono i contenuti

Questo è solo uno degli scenari possibili. Se al quadro si aggiungono le enormi potenzialità del linguaggio XSL, si comprende come l'epoca delle tante versioni di uno stesso sito sia davvero al termine.

Estensibilità

Dal momento che XHTML è XML diventa estensibile. Significa che sarà facilissimo incorporare in un documento parti scritte in uno dei tanti linguaggi della famiglia XML.
Ad esempio, inserire in una pagina delle formule matematiche complesse: basterà dichiarare il namespace relativo al linguaggio MathML e inserire nella pagina i tag specifici di quest'ultimo (codice tratto dal sito del W3C).

```
1  <html xmlns="http://www.w3.org/1999/xhtm" xml:lang="en" lang="en">
2  <head>
3          <title>A Math Example</title>
4  </head>
5  <body>
6          <p>The following is MathML markup:</p>
7          <math xmlns="http://www.w3.org/1998/Math/MathML">
8                  <apply> <log/>
9                          <logbase>
10                                 <cn> 3 </cn>
```

```
11                              </logbase>
12                    <ci> x </ci>
13                    </apply>
14           </math>
15  </body>
16  </html>
```

I frutti di questo approccio, che è il più semplice dei tanti possibili, sono ancora lontani dalla maturità. L'implementazione da parte dei browser è infatti carente, ma non mancano esempi funzionanti e di grande potenza. Parleremo in un'altra lezione di FML (Form Markup Language), un linguaggio che estende in maniera impressionante l'uso dei classici form di HTML.

Accessibilità

Una altro punto fondamentale è che i documenti scritti in XHTML e validati sono naturalmente più accessibili. Da una parte la validazione richiede l'uso obbligatorio di funzionalità come il testo alternativo per le immagini, dall'altra, una pagina che evita elementi non standard, ben definita nella struttura è di gran lunga più gestibile da browser alternativi come quelli vocali o testuali.

XHTML 2

XHTML 2, i cui scopi sarebbero dovuti essere il miglioramento del supporto ai dispositivi mobili e all'internazionalizzazione, non vedrà mai la luce. Il W3C aveva iniziato a lavorare su XHTML nel gennaio del 2000 e dopo nove anni il linguaggio che avrebbe dovuto essere una sorta di ponte fra HTML e XML, viene sospeso. L'obiettivo del consorzio è stato quello di focalizzare le risorse sullo sviluppo di HTML 5, per definire le specifiche di quello che dovrebbe diventare il nuovo standard universale del web. La definizione di uno standard nell'ambito del W3C passa attraverso gli stadi di Working Draft (bozza), Last Call (ultimo appello), Proposed Recommendation (proposta di raccomandazione) e Candidate Recommendation (raccomandazione candidata). Dal 2002 al 2009, il W3C ha rilasciato 8 Working Draft di XHTML e a dicembre dell'ultimo anno il processo di standardizzazione è stato interrotto visto che il Working Group dedicato alla specifica non ha più ragione di continuare tale progetto. Ian Jacobs, por-

tavoce del W3C, ha dichiarato che il consorzio si è progressivamente reso conto che XHTML 2 poteva essere destinato a un mercato molto più piccolo rispetto a quello di HMTL 5. I gruppi di lavoro dedicati ai due linguaggi, inoltre, lavorano da tempo a cose molto simili. In effetti, XHTML è quasi interamente incluso nelle specifiche di HTML 5 come serializzazione XML di HTML.

2 HTML5 : novità e tecnologie introdotte

L'HTML5 è un linguaggio di markup per la strutturazione delle pagine web. Proposto dal gruppo di lavoro WHATWG inizialmente in contrasto con il W3C per le lungaggini nel processo di evoluzione dello standard html e per la decisione del W3C di orientare la standardizzazione verso l'XHTML 2 che non garantiva retro compatibilità. Il W3C ha poi riconosciuto valide tali motivazioni, annunciando di creare un apposito gruppo per la standardizzazione dell'HTML5 e abbandonare l'XHTML 2.0[1]. Il W3C ha annunciato che la prima versione dello standard sarà pronta per il luglio 2014. Dal 2007 il WHATWG ha collaborato con il W3C in tale processo di standardizzazione, per poi decidere nel 2012 di separarsi dal processo di standardizzazione del W3C , creando di fatto due versioni dell'HTML5: la versione del WHATWG viene definita come "HTML Living Standard" e quindi in continua evoluzione, mentre quella del W3C sarà una unica versione corrispondente ad uno "snapshot" del Living Standard. Le **novità** introdotte dall'HTML5 sono finalizzate soprattutto a migliorare il disaccoppiamento tra struttura, definita dal markup, caratteristiche di resa (tipo di carattere, colori, e altro), definite dalle direttive di stile, e contenuti di una pagina web, definiti dal testo vero e proprio. Inoltre l'HTML5 prevede il supporto per la memorizzazione locale di grosse quantità di dati scaricati dal browser, per consentire l'utilizzo di applicazioni basate su web (come per esempio le caselle di posta di Google o altri servizi analoghi) anche in assenza di collegamento a Internet. Questo, come già ribadito, per una necessità crescente di poter utilizzare applicazioni sempre più complesse attraverso il web.

3 HTML5 per le applicazioni web strutturate

Con un'analisi delle features messe a disposizione da HTML5 si vuole ora mostrare come questa tecnologia si stia evolvendo verso l'idea di un web composto da applicazioni ben strutturate, garantendo strumenti all'altezza di tale compito:

- vengono rese più stringenti le regole per la strutturazione del testo in capitoli, paragrafi e sezioni; vengono introdotti elementi di controllo per i menu di navigazione; vengono migliorati ed estesi gli elementi di controllo per i moduli elettronici; vengono deprecati o eliminati alcuni elementi che hanno dimostrato scarso o nessun utilizzo effettivo; sostituzione del lungo e complesso doctype, con un semplice DOCTYPE html;
- vengono introdotti elementi specifici per il controllo di contenuti multimediali permettendone una gestione in modo nativo(tag video e audio);
- vengono estesi a tutti i tag una serie di attributi, specialmente quelli finalizzati all'accessibilità, finora previsti solo per alcuni;
- si ha l'introduzione dei WebSockets che danno una connettivita sempre maggiore, il che consente di effettuare chat in tempo reale, utilizzare giochi più veloci e migliorare la comunicazione tra client e server;
- viene introdotta la geolocalizzazione, dovuta ad una forte espansione di sistemi operativi mobili (quali Android e iOS, tra i più diffusi) attraverso le quali API si potranno effettuare altre importanti operazioni come l'accesso all'HW del dispositivo da parte delle applicazioni (webcam, contatti, calendario, giroscopio e altro);
- si ha un sistema alternativo ai normali cookie, chiamato Web Storage, più efficiente, il quale consente un notevole risparmio di banda;
- si avrà un offine e una migliore gestione dei dati, grazie a tutte le nuovi API per la gestione dei file, LocalStorage, App Cache, IndexedDB;
- vi è inoltre una standardizzazione di programmi JavaScript, chiamati Web Workers e la possibilità di utilizzare alcuni siti offline;

Inoltre in HTML5 confluiscono altre tecnologie oltre all'HTML inteso, in senso stretto, come semplice linguaggio di markup, che abbracciano diverse componenti dello sviluppo per il Web:

Semantica: come accennato grazie a un insieme più accurato di tag, il Web viene reso più fruibile, si aggiungono inoltre gli RDF (Resource Description Framework), i micro dati e i micro formati i quali permettono di dare un senso alla struttura in chiave semantica. I micro formati consentono alle pagine la possibilità di veicolare delle micro informazioni per diversi generi di contenuto: contatti, relazioni, eventi, prodotti, discussioni, ricette, e altro(tecnologie già supportate dalle precedenti versioni di HTML)

Canvas: Il canvas è un elemento HTML5 rappresentato dal tag "canvas". Può essere inteso come il corrispettivo digitale di una tela trasparente: uno spazio all'interno di una pagina web sul quale insistere con specifiche API adatte a tracciare linee, cerchi, rettangoli, immagini e altro ancora. Il canvas è, in estrema sintesi, una grande matrice di pixel, ognuno dei quali modificabile singolarmente nelle sue quattro componenti RGBA, rosso, verde, blu e alpha(la trasparenza). L'importanza di questo elemento HTML5 è enorme : esso da la possibilità di replicare i comportamenti fino ad oggi appannaggio della tecnologia Flash, tecnologia molto pesante ma allo stesso tempo indispensabile per una fetta enorme di servizi web.

CSS3 : presenta numerose novità riguardanti selettori, posizionamento, box model, gestione del testo, dei font, dei bordi, degli sfondi, dei colori, media query, trasformazioni, animazioni, e altro. La vasta scelta tra effetti, come trasparenze, ombreggiature, immagini di sfondo multiple e stili diversi che si possono applicare ai contenuti senza dover adattare il markup, consentono una personalizzazione grafica straordinaria.

3.1 Approfondimento sulle funzionalità

Si vogliono ora approfondire alcune delle nuove funzionalità fornite dall'HTML5 di particolare importanza per la creazione di applicazioni web strutturate.

Webworker

Una delle novità più interessanti è rappresentata dai WebWorkers. Grazie ad essi è possibile introdurre nell'ottica delle Web application la possibilità di supportare il concetto di concorrenza che sino ad ora non era realizzabile.L'esecuzione di codice javascript in una pagina Html prima portava ad una situazione di blocco fino alla fine dell'esecuzione dello script. Ciascun WebWorker rappresenta un entità che consente di eseguire codice JavaScript in un thread separato dal thread principale utilizzato per l'interfaccia. Ognuno di essi, viene identificato da un particolare file che implementa ciò che il WebWorker deve eseguire. Tali sotto-script vengono eseguiti in uno scope particolare, all'interno del quale non sarà possibile accedere agli elementi DOM della pagina, proprio per il fatto che essi devono agire indipendentemente dalla pagina e sono predisposti per eseguire operazioni complesse senza però bloccare la User Interface(UI), come invece potrebbe accadere in architetture che non sfruttano questa funzione.I WebWorker possono comunicare con l'ambiente esterno mediante lo scambio di messaggche avviene tramite l'utilizzo di una struttura orientata agli eventi e in particolare grazie al metodo postMessage è possibile inviare messaggi. Questi messaggi sono gestiti in modalità asincrona grazie alla proprietà onmessage associata al WebWorker. La limitazione che caratterizza i WebWorker consiste nell'impossibilità in alcun modo di poter accedere direttamente al DOM della pagina, cioè non è possibile modificare il markup della pagina tramite un WebWorker. Ciò non signi

ca che non è possibile interagire con la pagina principale tramite un WebWorker, ma è necessario rispettare degli appositi canali di comunicazione, nel nostro caso quelli fra pagina e WebWorker sono rappresentati dagli eventi. Inoltre i WebWorker non solo possono eseguire porzioni di codice in un contesto separato rispetto alla pagina principale, ma hanno la possibilità di creare e di utilizzare altri WebWorker e comunicare con essi con le stesse API utilizzate in precedenza per far comunicare la pagina principale con la WebWorker. Questa tecnica di delegazione da un lato permette di ottimizzare ancora di più gli script sfruttando al meglio i processori multicore ma dall'altro presenta notevoli difficoltà dal punto di vista dello sviluppo

e dell'integrazione tra i diversi componenti.

WebSockets

Una delle caratteristiche principali del web è l'architettura client-server, una modalità di comunicazione dove c'è un client, ben definito, che fa richieste ad un server, il quale fornisce le risposte quando è interrogato. Questa struttura è ottima per un web statico, fatto di pagine collegate tra loro in cui la richiesta di nuovo contenuto è stimolata dal click su un link da parte dell'utente del browser. Questa architettura non è adeguata ad applicazioni complesse e ricche di comunicazioni bidirezionali o dove le informazioni potrebbero dover arrivare quando sono disponibili e non quando l'utente fa qualcosa (ad esempio per applicazioni in real-time). HTML5 risolve in maniera efficiente questo problema offrendo agli sviluppatori una soluzione semplice ed efficace: le WebSockets API che introducono, una funzionalità tra le più attese: la possibilità di stabilire e mantenere una connessione dati (asincrona) tra browser e server remoto sulla quale far transitare messaggi in entrambe le direzioni. Le API offrono un semplice meccanismo grazie all'oggetto WebSocket, al metodo send e all'evento onmessage (sopra descritti). La creazione di un nuovo WebSocket richiede come unico parametro obbligatorio l'url verso la quale si vuole stabilire la connessione. Il protocollo può essere ws o wss, dove il secondo indica la richiesta di una connessione sicura:

```
1 var webSocket = new WebSocket('ws : //echo:websocket:org/');
```

Questa riga permette di creare una connessione (socket) al server con cui abbiamo deciso di instaurare il nostro dialogo. Una volta creato l'oggetto si hanno a disposizione due metodi per l'invio di informazioni e per la chiusura del canale di comunicazione:

```
1 webSocket.send(''Hello world"");
2 webSocket.close()
```

Invece si utilizzano i listener per interagire con i messaggi dal server:

```
1 webSocket.onopen = function(event) {
2 console.log("Connection opened!");
3 }
4 webSocket.onmessage = function(event) {
```

```
5  console.log("Server says " + event.data);
6  }
7  webSocket.onerror = function(event) {
8  console.log("Error!!");
9  }
10 webSocket.onclose = function(event) {
11 console.log("Connection closed!");
12 }
```

Tra le soluzioni degne di nota che implementano tale strumento sppiccano Pusher, un servizio che offre la possibilità di gestire eventi real-time attraverso l'utilizzo di WebSocket, e altrettanto valido jsTerm, un'applicazione che consente di collegarsi a server remoti utilizzando un proxy, scritto in node.js, tra il protocollo WebSocket e Telnet.

Applicazioni web offine e LocalStorage

La progettazione di applicazioni Web in grado di funzionare senza un accesso a Internet appare una contraddizione, soprattutto perchè come detto nei capitoli precedenti tutto ruota attorno al Web. In ogni caso è un fatto innegabile che è praticamente impossibile che una connessione alla rete Internet sia sempre disponibile e non abbia mai interruzioni. Pertanto per far fronte a tale eventualità, le nuove specifche Html5 hanno introdotto nuove API per gestire queste caratteristiche: viene fornita una nuova funzionalità di caching per supportare le applicazioni web in modalità offine. Può accadere infatti che la connessione alla rete possa non essere momentaneamente disponibile e per far fronte a tale eventualità è stata elaborata una parte della specifica di HTML5 denominata OfflineWebApplication che consente di progettare applicazioni web utilizzabili quando la connessione di rete non è attiva e consiste nella possibilità di far caricare in una cache locale, denominata application cache, dei file di risorse rappresentati da pagine web, filmati Flash, immagini, fogli di stile CSS, Javascript e così via. Il file ha questo formato:

```
1
2  CACHE MANIFEST
3  index.html
4          /images/logo.png
5          /css/styles.css
6          /js/jquery-1.4.min.js
```

```
7          /js/offline.js
```

che verrà poi caricato nell'Html attraverso il tag

```
1  <html manifest="offline.manifest">
```

Quando visiteremo il sito in questione il browser chiederà se autorizziamo l'utilizzo della cache e a quel punto i contenuti che abbiamo specifcato nel file manifest verranno salvati sul nostro computer così da permetterci una navigazione completa del sito anche quando siamo offine. Molto importante è che il file manifest sia servito dal server web al browser con il MIME type giusto; in caso contrario il browser non lo riconoscerà come manifest per la cache. E necessario fare attenzione al fatto che il semplice aggiornamento di un file incluso nel manifest non comporta l'aggiornamento della versione in cache pertanto man mano che andremo ad aggiungere risorse al progetto (immagini, librerie, etc...) sarà necessario includerle anche nel file manifest. Il vantaggio principale rispetto alla cache tradizionale è il controllo: mentre solitamente è il browser a decidere quali file memorizzare in cache, con l'application cache possiamo dire noi quali risorse tenere in memoria così da offrire agli utenti un'esperienza di navigazione offine completa. Inoltre l'application cache consente di salvare file che non sono stati visitati ma che potrebbero essere utili per una navigazione completa del sito. Nella normale cache del browser una pagina per essere memorizzata deve essere stata visitata almeno una volta. Inoltre molto importante è l'Offline Storage (o anche Local Storage)il quale effettua la cattura di specifici dati generati dall'utente oppure di risorse per cui l'utente ha mostrato interesse; una chiave in questa istanza ha visibilità a livello di dominio, i dati conservati nell'istanza sono quindi visibili in tutte le finestre aperte sul dominio. Local Storage è un'archiviazione persistente. L'utilizzo è molto semplice, si usano i metodi setItem e removeItem per inserire e cancellare item utilizzando un meccanismo chiave valore. Ad esempio per inserire un valore

```
1  localStorage:setItem(''keyName"", ''keyValue"");
```

Rispetto ai cookies il Web Storage è la soluzione ideale per progetti in cui abbiamo bisogno di salvare lo stato di un'applicazione e i dati trasmessi sono pesanti: il limite di 5 Mb consente di avere un database piuttosto ampio; nel caso in cui dovessimo superare i limite al nostro tentativo di

aggiungere altri dati al localStorage riceveremo un errore QUOTA EXCEEDED ERR. I cookies, inoltre, sono parte del protocollo HTTP per cui i dati che contengono vengono inviati ad ogni richiesta, creando un notevole traffico.
Il local storage è uno strumento molto potente che, combinato con le offline web applications , consente di creare delle vere e proprie applicazioni indipendenti dalla effettiva connessione ad Internet o meno. L'attuale utilizzo più valido è sui siti web pensati per il mobile network, consentendo agli sviluppatori di creare di fatto delle vere a proprie applicazioni funzionanti anche quando non è presente una connessione di rete.

In definitiva l'HTML 5 nasce per risolvere i problemi legati al crescere della complessità delle applicazioni web, proponendo agli sviluppatori un linguaggio capace di adattarsi alle nuove necessità, sia dal punto di vista della strutturazione del contenuto sia da quello del supporto a nuovi linguaggi più prestanti.

Capitolo 3

Applicazioni Web Strutturate : Dart come caso di studio

1 Dart

Dart brings structure to web app engineering
with a new language, libraries, and tools.
- Dart official site -

Dart è un interessante progetto nato da GOOGLE per creare un linguaggio cross-platforms e cross-browser funzionante sia lato server che lato client che permetta di sviluppare applicazioni e strutturate. È stato presentato alla conferenza GOTO Aarhus 2011 dal 10 al 12 Ottobre 2011. L'obiettivo immediato è quello di sostituire Javascript (che è un linguaggio oggettivamente molto complesso da debuggare) e in futuro anche i linguaggi server side permettendo di utilizzare un solo linguaggio per l'intera applicazione. A presentarlo sono stati i due ingegneri che lo hanno creato, Gilad Bracha e Lars Bak, il primo è tra i creatori di Java, il secondo è l'ideatore di V8, la tecnologia che dà al browser Chrome la velocità che lo caratterizza. Come spiega l'azienda, Dart nasce per aiutare gli sviluppatori a creare un linguaggio per la programmazione web strutturato ma flessibile, facendolo apparire subito familiare e naturale ai programmatori i quali potranno realizzare applicazioni capaci di offrire prestazioni elevate su qualsiasi browser moderno. Il codice è stato progettato per essere

simile a linguaggi già esistenti, come C# e Java, ma con la necessaria essibilità dei linguaggi web-oriented; inoltre è lampante quanto questo nuovo lunguaggio abbia in comune con Javascript.
I progetti sviluppati potranno essere eseguiti con virtual machine DartVM che al momento è presente solo in una versione Alpha di Google Chrome(Dartium); Dart potrà essere eseguito all'interno del codice HTML con l'apposito MIME type 'application/dart' da utilizzare nei tag ¡ script ¿ per includere il sorgente del linguaggio. Per il momento è possibile utilizzare questo linguaggio grazie a un compilatore (Dart Cross Compiler) che traduce il codice Dart in codice Javascript, il quale può essere eseguito all'interno del browser, in modo da permettere sia agli sviluppatori di prendere confdenza con l'ambiente, sia al linguaggio di diffondersi. L'intero progetto è open-source e sul sito uffciale è possibile scaricare gratuitamente un set di librerie di base e i tool preliminari per la compilazione e l'esecuzione di codice Dart. Il progetto è in fase iniziale e Google mira a raccogliere esperienze e feedback dagli sviluppatori per capirne le reali potenzialità.

2 Perchè analizzare Dart

Come già accennato nel primo capitolo, al momento il web necessita di un linguaggio di programmazione strutturato .Col passare del tempo, il Web ha dovuto far fronte alla sempre più crescente richiesta di integrazione di funzionalità interattive e contenuti multimediali.Per fare ciò sono state sviluppate numerose tecnologie che di volta in volta hanno permesso l'integrazione delle funzionalità reputate maggiormente significative per l'evoluzione del Web; però nel momento in cui si fanno interagire queste tecnologie le une con le altre, si assiste ad una perdita della percezione modulare dell'applicazione e ad un aumento esponenziale della complessità dei controlli per il debugging, nonchè ad un appesantimento non trascurabile delle operazioni di caricamento. Javascript ad esempio permette lo sviluppo semplice e veloce di piccole applicazioni, ma è altamente sconsigliato nel caso in cui l'applicazione da realizzare sia mediamente complessa. La mancanza di una tipizzazione rende, infatti, molto diffcile la gestione del codice Javascript, rendendo di fatto impossibile la realizzazione di validi

tool di programmazione o di un supporto per moduli, package o librerie.
La necessità di un approccio strutturato dei linguaggi per il web, sia per gestire effcacemente la realizzazione di grandi applicazioni, sia per far fronte ad un Web sempre in rapida evoluzione, ha portato all'ideazione di un linguaggio che incorporasse tutte le funzionalità sino ad ora gestite da tecnologie diverse ed eterogenee, che risultasse familiare ai programmatori e che fornisse quel supporto strutturale che permette una buona ingegnerizzazione delle applicazioni.
Dart nasce anche con l'obiettivo di risolvere alcuni dei problemi che si presentano agli sviluppatori Web:

- spesso piccoli script evolvono senza controllo in applicazioni web di grandi dimensioni che non hanno una struttura ben ordinata e comprensibile. Queste applicazioni monolitiche non possono essere suddivise per far sì che diverse squadre di sviluppatori si concentrino sulla programmazione contemporanea di più parti dell'applicazione in modo indipendente.Inoltre è difficile essere produttivi quando un'applicazione web diventa grande.
- il successo dei linguaggi di scripting è dovuto principalmente alla loro natura semplice che permette di scrivere codice velocemente; ciò comporta che generalmente lo scambio di informazioni tra differenti parti del sistema sia improvvisato e descritto in commenti all'interno del codice, piuttosto che essere definito nella struttura del linguaggio stesso, rendendo arduo se non impossibile a chiunque non sia l'autore del codice, capire e mantenere un particolare pezzo di programma.
- i linguaggi esistenti costringono lo sviluppatore a scegliere tra linguaggi statici o dinamici. I tradizionali linguaggi statici richiedono uno stile di programmazione rigoroso che di solito viene percepito troppo stringente ed eccessivamente vincolato.
- gli sviluppatori non sono mai stati in grado di creare sistemi omogenei che comprendano sia client che server, ad eccezione di pochi casi, come Node.js e Google Web Toolkit (GWT).
- linguaggi e format diversi comportano cambi di contesto che sono ingombranti e aggiungono complessità al processo di codifica

3 Caratteristica del linguaggio

Le caratteristiche principali del linguaggio Dart si focalizzano sulle classi, la tipizzazione opzionale, la librerie e tooling.

3.1 Classi e Interfacce

Classi e interfacce forniscono un meccanismo chiaro per definire in modo efficiente le API. Questi costrutti consentono l'incapsulamento e il riutilizzo di metodi e dati fornendo un insieme di blocchi riutilizzabili ed estensibili.

- un'interfaccia definisce un set di metodi e costanti, a volte ereditando da altre interfacce.
- una classe può implementare più interfacce, ma eredita solo da una singola superclasse.

L'esempio seguente definisce un'interfaccia, insieme a una classe e sotto-classe che lo implementano:

```
1  interface Shape {
2     num perimeter();
3  }
4  class Rectangle implements Shape {
5     final num height, width;
6     // Compact constructor syntax.
7     Rectangle(num this.height, num this.width);
8     // Short function syntax.
9     num perimeter() => 2*height + 2*width;
10 }
11
12 class Square extends Rectangle {
13    Square(num size) : super(size, size);
14 }
```

Una **classe** definisce la forma e il comportamento di un insieme di oggetti che sono le sue istanze; è dotata di costruttori, metodi, getter, setter e variabili. A parte i costruttori, gli altri membri possono essere sia membri di istanza che membri statici.

Si ricorda che l'attuale javascript non possiede tale struttura sintattica: infatti è proprio la presenza delle Classi che fa di Dart un linguaggio più

adatto alla strutturazione delle applicazioni; con javascript si dovrebbe in alternativa utilizzare una funzione del tipo :

```
<script type="text/javascript">
var rettangolo = new function(param_height, param_whidth)
{
    this.height = param_height;
    this.whidth = param_whidth;
    this.perimetro = function()
    {
        return 2 * this.height + 2 * this.width;
    }
}
</script>
```

In javascript, come detto, si utilizzano i prototipi (objects prototypes): ogni tipo di oggetto ha una proprietà che è possibile estendere o ereditare:

```
<script>

Function.prototype.inherits = function(superclass) {
  var temp = function() {};

  temp.prototype = superclass.prototype;
  this.prototype = new temp();
}

</script>
```

Con questa funzione basta scrivere ClasseFiglia.inherits(ClassePadre) per creare di fatto una ereditarietà. Tramite questa funzione è possibile inoltre invocare il costruttore padre all?interno del costruttore figlio semplicemente chiamando il nome della classe con i parametri necessari.
Spesso quando si utilizza tale costrutto in javascipt, non si immagina che qualcuno potrà poi estenderla, e quindi non si implementa il codice in questa ottica. L'esempio sopra ne è una prova in quanto in quella situazione è un pò difficile aggiungere nuove funzionalità alla DataGrid mantenendone la struttura interna. Questo è uno dei principali motivi per cui si sente la necessità di un nuovo linguaggio come Dart.

Si noti inoltre l'assenza delle interfacce: nonostante ciò, grazie al grande sviluppo e utilizzo del javascript, alcuni programmatori hanno messo appunto procedimenti per emulare tale costrutto, ma con risultati insoddi-

sfacenti, poco intuitivi e di difficile utilizzo.Se ne riporta ora un'esempio. Per prima cosa si dovrebbe includere nell'html un file "Interface.js";

```
1
2  var Interface = function (objectName, methods) {
3
4   // Check that the right amount of arguments are provided
5   if (arguments.length != 2) {
6     throw new Error ("Interface constructor called with "
7     + arguments.length + "arguments, but expected exactly 2.");
8
9  }
10
11  // Create the public properties
12  this.name = objectName;
13
14  this.methods = [];
15
16  // Loop through provided arguments and add them
17  //to the 'methods' array
18  for (var i = 0, len = methods.length; i < len; i++) {
19
20          // Check the method name provided is written as a String
21          if (typeof methods[i] !== 'string') {
22          throw new Error ("Interface constructor expects
23           method names to be " + "passed in as a string.");
24          }
25          // If all is as required then add the provided method
26          // name to the method array
27          this.methods.push(methods[i]);
28
29          }
30 };
31
32 /*
33  * Adds a static method to the 'Interface' constructor
34  * @param object | Object Literal | an object literal
35  * containing methods that should be implemented
36  */
37 Interface.ensureImplements = function (object) {
38
39  // Check that the right amount of arguments are provided
40  if (arguments.length < 2) {
41                  throw new Error ("Interface.ensureImplements
42                   was called with "+ arguments.length +
43                   "arguments,  but expected at least 2.");
44
45          }
46
47          // Loop through provided arguments
48          // (notice the loop starts at the second argument)
49          // We start with the second argument on purpose
```

```
50          // so we miss the data object
51
52   for (var i = 1, len = arguments.length; i < len; i++) {
53
54    // Check the object provided as an argument is an instance
55    // of the 'Interface' class
56    var interface = arguments[i];
57    if (interface.constructor !== Interface) {
58    throw new Error ("Interface.ensureImplements expects
59           the second argument to be an instance of the
60                                  'Interface' constructor.");
61                  }
62
63    // Otherwise if the provided argument IS an instance
64    // of the Interface class then loop through provided
65    // arguments (object) and check they implement
66    // the required methods
67
68    for (var j = 0, methodsLen = interface.methods.length;
69                                            j < methodsLen; j++) {
70
71          var method = interface.methods[j];
72
73          // Check method name exists and that it is a
74          // function (e.g. test[getTheDate])
75          // if false is returned from either check
76          // then throw an error
77
78        if (!object[method]
79              || typeof object[method] !== 'function') {
80
81          throw new Error ("This Class does not implement the '"
82          + interface.name + "' interface correctly. The method '"
83                                    + method + "' was not found.");
84          }
85    }
86  }
87  };
```

In seguito nell'Html avremo un scropt per la creazione di un'istanza di tale interfaccia per passare poi i metodi di cui si necessita.

```
1
2  // We pass into the Interface the name of the current
3  // Interface instance, followed by an Array of the
4  // methods we are expecting to find
5
6  var test = new Interface('test', ['details', 'age']);
7
8  var properties = {
9    name: "Mark McDonnell",
10    actions: {
```

```
11       details: function() {
12        return "I am " + this.age() + " years old.";
13     },
14
15     age: (function(birthdate) {
16         var dob = new Date(birthdate),
17
18         today = new Date(),
19         ms = today.valueOf() - dob.valueOf(),
20
21         minutes = ms / 1000 / 60,
22         hours = minutes / 60,
23         days = hours / 24,
24         years = days / 365,
25
26         age = Math.floor(years)
27       return function() {
28          return age;
29                  };
30          })("1981 08 30")
31
32         }
33 };
34
35 // Create a Person constructor that will implement the
36 // above properties/methods
37 function Person(config) {
38
39          // Pass in the methods we are expecting,
40          // followed by the name of the Interface
41          //instance that we're checking against
42          Interface.ensureImplements(config.actions, test);
43
44
45          this.name = config.name;
46          this.methods = config.actions;
47
48 }
49
50 // Create a new instance of the Person constructor...
51 var me = new Person(properties);
52
53
54 // ...and make sure the methods are working
55 alert(me.methods.age());
56 alert(me.method.details());
```

Si elencano ora alcune caratteristiche della Classe in Dart:

- ha una e una sola super-classe, fatta eccezione per la classe Object che non ha super-classe;

- può implementare una o più interfacce tramite la clausola implements;
- viene detta astratta se è definita esplicitamente tramite il modificatore abstract oppure se contiene almeno un metodo astratto;
- non può dichiarare due membri con lo stesso nome, tranne nel caso in cui tali membri siano un getter e un setter entrambi membri di istanza o entrambi membri statici. Ovviamente non è possibile definire metodi setter per una variabile final.

I costruttori di una classe possono essere di tre tipi:

1. generativi: un costruttore generativo è un metodo denominato secondo il nome della classe cui appartiene e viene invocato tramite la parola chiave ”new“, restituendo un'istanza della classe.

```
1  class A {
2    var x;
3    A([x]): this.x = x;
4  }
5
6  o anche
7
8  class A {
9    var x;
10   A([this.x]);
11 }
```

2. fabbriche: una fabbrica è un metodo statico, identificato dal modificatore factory, denominato secondo il nome della classe cui appartiene e, a differenza di un costruttore generativo, può restituire anche istanze di classi del sottotipo del valore di ritorno.
3. costanti: un costruttore costante, identificato dal modificatore const, viene utilizzato all'interno di classi di solo valori final per generare oggetti costanti in fase di compilazione.

```
1  class C {
2    nal x;
3    nal y;
4    nal z;
5    const C(p, q): x = q, y = p + 100, z = p + q;
6  {
```

Un'interfaccia definisce il modo con cui si interagisce con un oggetto; ha metodi, getter e setter, costruttori e opzionalmente anche un set di super-interfacce. Non è possibile specificare alcun valore di default nella firma di un membro dell'interfaccia. In generale, un interfaccia rappresenta una sorta di "promessa" che una classe si impegna a mantenere. La promessa è quella di implementare determinati metodi di cui viene resa notala definizione . Ciò che è importante non è tanto come verranno implementati tali metodi all'interno della classe ma, piuttosto, che la denominazione ed i parametri richiesti siano assolutamente rispettati. Al suo interno è possibile definire degli operatori e una classe fabbrica da utilizzare quando si cerchi di reificare una istanza richiamando il costruttore attraverso l'interfaccia. Per le interfacce valgono le stesse regole di controllo statico o a compile-time che valgono per le classi.

```
Definizione di un'interfaccia

interface Hashable {
  int hashCode();
}

e successiva implementazione

class Point implements Hashable {
  num x, y;
  ...

  // Required by Hashable.
  int hashCode() {
    int result = 17;
    result = 37 * result + x.hashCode();
    result = 37 * result + y.hashCode();
    return result;
  }

  // Always implement operator == if the class implements Hashable.
  bool operator==(other) {
    if (other == null) return false;
    if (other === this) return true;
    return (other.x == x && other.y == y);
  }
}
```

In Dart spesso è possibile creare oggetti direttamente da un'interfaccia, invece di dover trovare una classe che implementa tale interfaccia. Questo è possibile perchè molte interfacce hanno una factory class, una classe

che crea oggetti che implementano l'interfaccia. Per esempio, se il codice dice newDate:now(), la classe fabbrica per l'interfaccia di Data crea un oggetto che rappresenta l'ora corrente.

3.2 Generici

Dart supporta i generici. Una classe, un'interfaccia o un membro d'istanza possono essere generici, utilizzare cioè parametri formali che sottintendono una famiglia di dichiarazioni, una per ogni set di parametri veri e propri forniti nel programma. Un parametro può essere seguito dalla clausola extends, per indicare un upper bound del parametro; se la clausola extends non è presente, è sottinteso che la super-classe sia Object. Nel caso in cui si cerchi di definire un upper bound per un parametro che è un supertipo di quello stesso upper bound, si genera un warning statico.

3.3 Isolate

Proprio come JavaScript utilizza i WebWorker, introdotti dalla nuova specifica Html5 discussa nel capitolo precedente, Dart utilizza gli Isolate per supportare il concetto di concorrenza. Gli Isolate sono le unità di concorrenza utilizzate da Dart e si basano sul modello ad attori. Il modello ad attori è un modello di calcolo concorrente basato sugli attori intesi come l'entità primaria. Il modello ad attori adotta la filosofia ogni cosa è un attore; tale concetto è simile al tutto è un oggetto, filosofia utilizzata dai linguaggi di programmazione orientata agli oggetti, la quale però differisce in quanto un software object-oriended solitamente viene eseguito in modo sequenziale, mentre il modello ad attori è intrinsecamente concorrente. Dato che ogni cosa è un attore, la comunicazione fra gli attori avviene utilizzando il modello message passing. Un attore è un'entità computazionale che, in risposta ad un messaggio ricevuto, può contemporaneamente:

- inviare un numero finito di messaggi ad altri attori;
- creare un numero finito di attori;
- determinare il comportamento da utilizzare per il successivo messaggio che riceve.

L'unico meccanismo disponibile per far comunicare gli Isolate fra di loro è tramite lo scambio di messaggi. I messaggi vengono inviati attraverso le ports che sono RecivePort o SendPort; sono gli elementi che stanno alla base di un canale di comunicazione per gli Isolate. La prima classe, cioè ReceivePort, rappresenta il ricevitore, invece SendPort rappresenta il mittente all'interno del canale di comunicazione; ogni Isolate quando viene creato possiede come impostazione predefinita una porta di ricezione. Questa ReceivePort viene creata automaticamente e viene comunemente usata per stabilire la prima comunicazioni tra gli Isolate:

```
1  interface Isolate{
2    ...
3    ReceivePort get port()
4    ...
5  }
6
7  interface ReceivePort{
8    ...
9    void close()
10   void receive(void callback(message, SendPort replyTo)
11   SendPort toSendPort()
12 }
```

Per la creazione degli Isolate sono disponibile due tipologia di API: ”spawnFunction“ e ”spawnUri“: la prima consente la creazione di un Isolate, il quale utilizza il codice passato come argomento con codice del corrente Isolate; la seconda crea e genera un isolate il cui codice è disponibile all'uri passato come parametro. Entrambi creano un Isolate avente di default una ReceivePort, in più entrambe restituiscono una SendPort utilizzabile per l'invio dei messaggi all'Isolate. A differenza dei WebWorker in JavaScript, gli Isolate non sono obbligatoriamente contenuti all'interno di un file esterno, ma possono essere presenti all'interno del codice. Questo è possibile grazie alle classi che definiscono gli Isolate.
Le API che definisco gli Isolate sono aggiornamento. Nuove API saranno presto aggiunte mentre altre saranno rimosse.Nel prossimo futuro si sta già parlando di aggiunge un'API per creare DOM Isolate ovvero Isolate con l'accesso al DOM, i quali, si prevede, saranno messi in esecuzione sul thread UI.Ad ogni modo si pensa che presto, data l'importanza dell'argomento, Dart fornisca API ben definite per la creazione di applicazioni basate sulla concorrenza, in modo da rendere l'interazione con l'utente e

le attività di computazione operazioni parallele.

3.4 Tipizzazione opzionale

Una delle caratteristiche più innovative del linguaggio di programmazione Dart rispetto alle alternative per il web, è l'uso di tipi opzionali. Prendendo in considerazioni i tipi di dati, nell'insieme dei linguaggi di programmazione ve ne sono alcuni, come i linguaggi macchina della maggior parte degli elaboratori detti linguaggi non tipizzati in quanto non prevedono tipi di dati o in ogni caso consentono l'utilizzo di un unico tipo, cioè una configurazione di bit, contenente tutti i valori possibili. Oltre ai linguaggi non tipizzati, vi sono i linguaggi tipizzati: in questi linguaggi di programmazione è necessario associare alle variabili, o alle espressioni delle annotazioni o dichiarazioni di tipo. In base al linguaggio di programmazione che si usa, queste annotazioni di tipo, devono essere specificate esplicitamente dal programmatore oppure possono essere generate in modo automatico dall'interprete o dal compilatore.Dart è un linguaggio a tipizzazione dinamica, cioè è un linguaggio di programmazione dove è possibile scegliere se scrivere programmi che utilizzano o meno le annotazioni di tipo. I programmatori Dart possono aggiungere opzionalmente al loro codice tipi statici. Secondo le preferenze del programmatore e lo stadio dell'applicazione, il codice può migrare da un semplice non tipato e sperimentale prototipo a una complessa applicazione modulare con tipi che, come sarà sottolineato ne fa un grande strumento per lo sviluppo di applicazioni strutturate grazie al type Checking. Poichè i tipi stabiliscono l'intento del programmatore, serve meno documentazione per spiegare cosa succede nel codice e si possono usare strumenti di type checking per il debugging. Dart fornisce, a scelta del programmatore, un mix di verifiche statiche e dinamiche. Durante la sperimentazione, il programmatore può scrivere codice senza tipo per la prototipazione semplice. Via via che l'applicazione diventa più grande e più stabile, i tipi possono essere aggiunti per aiutare il debug e imporre una struttura dove desiderato.

3.5 Librerie

Gli sviluppatori possono creare e utilizzare librerie che sono garantite non cambiare durante il runtime. Pezzi di codice sviluppato in modo indipendente possono quindi contare su librerie condivise. Dart fornisce due set di librerie a supporto della programmazione:

- **DOMLibrary** - contiene le interfacce per il DOM HTML5, basato liberamente sullo standard HTML5 specificato dal W3C/WHATWG.
- **CoreLibrary** - contiene interfacce per supportare strutture dati comuni e operazioni.

La gerarchia dell'interfaccia della Core Library, attualmente ha tre parti principali che estendono le seguenti interfacce di base:

- **Iterable** - L'interfaccia Iterable permette di ottenere un Iterator da un oggetto Iterable; l'interfaccia viene utilizzato nei costrutti for-in per iterare su un oggetto Iterable; Collection è un interfaccia che implementa Iterable e definisce metodi implementati dagli interfaccia Set, List e Queue; HashSet è un'implementazione più specifica di Set.
- **Map** - l'interfaccia Map è un contenitore associativo che associa una chiave ad un valore; l'interfaccia HashMap non garantisce un ordine particolare delle chiavi o dei valori restituiti dall'utilizzo di getKeys e getValues.
- **Comparable** - l'interfaccia è implementata da Date, Duration, num e String.
- **Hashable** - l'interfaccia e implementata da num e String.
- **Pattern** - l'interfaccia è implementata da String e RegExp.

Molto importante è la libreria che Dart mette a disposizione allo scopo di supportare la computazione lato server.Infatti è possibile eseguire una VM in grado di eseguire servizi anch'essi scritti in Dart; nella libreria sono incluse API per la gestione del file system, di stream di dati, socket, processi e thread. La VM è in grado di gestire le più comuni funzionalità di un'applicazione distribuita di tipo client-server come ad esempio l'elaborazione di richiesta HTTP o il recupero di dati inviati tramite socket stream.

Dart includeà un set corposo di ambienti di esecuzione, librerie e strumenti di sviluppo costruiti per supportare il linguaggio. Questi strumenti permetteranno uno sviluppo produttivo e dinamico, includendo debug cambia-e-continua e oltre, sino a raggiungere uno stile dove ”tu programmi lo schizzo di un’applicazione, lo esegui e riempi i buchi mentre esegui“.

4 Dart per applicazioni web strutturate

Dart supports classes as a fundamental structural building block for libraries and apps.Classes define the structure of an object, and you can extend them to create more specialized definitions. New features such as implicit interfaces and named constructors, make it easier to say more while typing less.
- Dart official site -

Una delle caratteristiche più importanti che rendono Dart una buona scelta per costruire software strutturato sta nella presenza delle **interfacce** e delle **classi** già presentate in precedenza. Al contrario di JavaScript che è un linguaggio di scripting Object-Based, ovvero un linguaggio che si basa sugli oggetti e non prevede l’utilizzo delle classi, Dart utilizza questo costrutto per la creazione degli oggetti: un oggetto creato da una certa classe viene detto istanza di tale classe. Le interfacce sono tipi che dicono come interagire con il relativo oggetto che verrà istanziato. Dell’interfaccia si possono specificare metodi, costruttori, getters e setters, e superinterfacce: grazie a questo strumento, senza specificare il codice interno, il programmatore acquisisce tutte le informazioni necessarie per l’utilizzo di un particolare tipo di dato creato in precedenza e, cosa più importante, può strutturare il proprio progetto attraverso un livello di astrazione superiore: può creare solamente l’interfaccia dei componenti necessari, per poi lasciare l’implementazione di questi ad altri team di sviluppo, permettendo quindi una suddivisione dei compiti; tutto ciò fa parte della logica della progettazione strutturata. Creare un software, soprattutto se di grandi dimensioni, richiede un iniziale processo ingegneristico di studio dei requisiti e delle problematiche da affrontare: Dart risulta un’ottimo strumento per

passare alla fase successiva di sviluppo grazie appunto a questi strumenti.

Come già affermato, Dart permette una **tipizzazione opzionale**: questa decisione probabilmente è stata fatta per poter permettere a chi viene dal mondo javascript di continuare a programmare senza una tipizzzione delle variabili; tuttavia se si decide di strutturare il proprio codice utilizzando una politica di tipizzazione ben definita. si acquissce una potenza di controllo del codice molto superiore grazie al type checking, operazione che non è possibile effettuare ad esempio in javascript: si tratta di verificare e stabilire se i valori assegnati ad una variabile o le operazioni eseguite su di essa siano di un tipo ammissibile per il tipo della variabile, ovvero permette di verificare se i vincoli imposti dai tipi siano soddisfatti.

Altra caratteristica di questo linguaggio che risulta molto importante nella creazione di applicazioni web è l'**integrazione di Dart con HTML5**: i tag script di HTML forniscono un attributo type per definire il linguaggio dello script. Per Dart, questo attributo ha il valore ”application/dart“. Come per altri tag per script, il contenuto può essere inserito come corpo del tag script o specificato con un URL utilizzando l'attributo src. Lo script Dart dovrà avere una funzione di primo livello main() o dichiarata direttamente nello script o in un file importato: il browser invoca il main() al caricamento. Lo script Dart potrà avere ulteriori comandi source e import per includere altri script o librerie. Incorporare il codice Dart è differente da incorporare codice Javascript. Ogni pagina HTML può avere più tag script Dart, ma ogni tag script nella pagina viene eseguito isolatamente. In Javascript, le dichiarazioni in ogni tag vengono combinate nello stesso namespace. In Dart il codice all'interno di un tag script non può accedere al codice in un altro tag a meno di utilizzare source o import ed ogni script deve avere il proprio main() perchè possa essere eseguito. . Il codice Dart viene eseguito solo dopo che la pagina è stata elaborata: i programmatori Dart, quindi, possono assumere che il DOM sia caricato completamente.

Inoltre c'è un aspetto molto importante riguardante **la gestione degli Eventi**: in Javascript, i programmatori devono inserire ascoltatori di eventi direttamente nell' HTML; con Dart nonostante questo sia permesso, è scorag-

giato, poichè la pagina HTML viene caricata molto più velocemente se gli ascoltatori sono aggiunti in seguito separatamente: infatti mette a disposizione la possibilità di gestire gli eventi all'interno dello stesso codice Dart. Si può in questo modo semparare ulteriormente la logica del programma dalla sua dipendenza alla tecnologia Html su cui si appoggia, permettendo una progettazione più accurata del software.

4.1 Alcuni esempi di studio

Si vedranno ora alcuni esempi di codice riguardanti ciò che è stato appena descritto.

Classi e Interfacce

Se si pensa al processo di creazione di un software attraverso una metodologia ingegneristica, si avrà un susseguirsi di fasi che porteranno lo sviluppatore dall'idea iniziale del prodotto sino alla sua realizzazione. Si considera ad esempio di voler realizzare una semplice applicazione per il raddoppio data una stringa. Si inizierà formulando un'interfaccia che soddisfi i nostri requisiti:

```
1
2 interface doubler {
3   String doubleAsString(String) {
4   }
5 }
```

Successivamente in base alla necessità si potranno creare diverse classi che soddisfano tali richieste implementando questa iniziale interfaccia: ad esempio se si necessità di un componente che data una stringa, la converta in un numero e poi ne raddoppi il valore si avrà numDoubler:

```
1 class numDoubler implements doubler {
2   String doubleAsString(String input) {
3     double number = 0.0;
4     try {
5       number = Math.parseDouble(input);
6     }
7     catch (BadNumberFormatException ex) {
8       return ('');
9     }
```

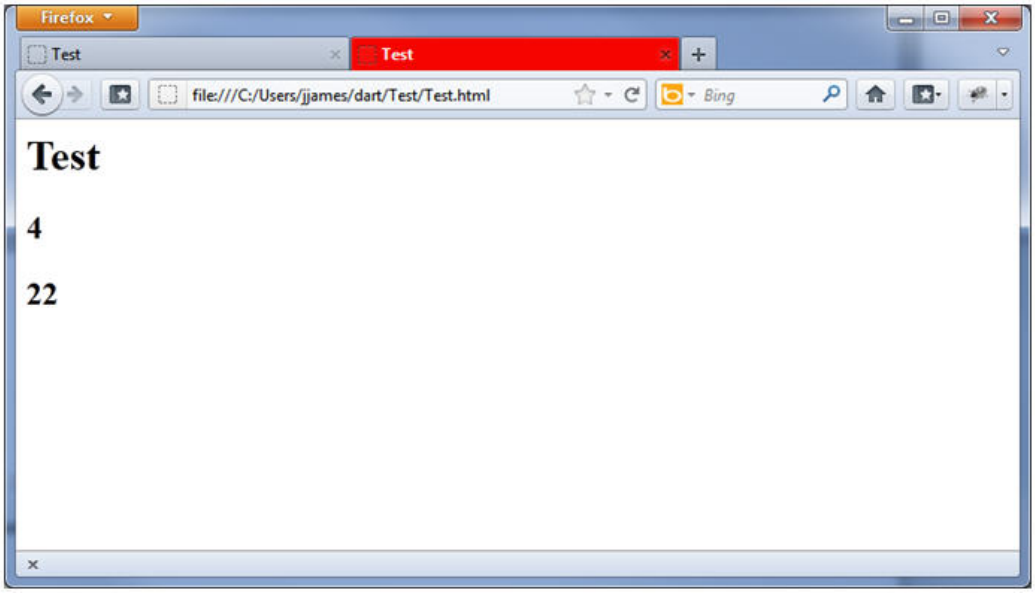

Figura 3.1: Doubler Example

```
10      return (number * 2).toString();
11    }
12  }
```

Invece per un componente che data una stringa ne concateni una sua copia si avrà stringDoubler:

```
1  class stringDoubler implements doubler {
2    String doubleAsString(String input) {
3      return input + input;
4    }
5  }
```

Complentando quindi attraverso un main

```
1  void main() {
2    var numDub = new numDoubler();
3    var strDub = new stringDoubler();
4    document.query("#doubledNum").innerHTML
5                            = numDub.doubleAsString('2');
6    document.query("#doubledString").innerHTML
7                            = strDub.doubleAsString('2');
8  }
```

e la relativa integrazione con l'Html si avarà :

```
1  <html>
2    <head>
3      <title>Test</title>
4    </head>
```

```
5     <body>
6       <h1>Test</h1>
7       <h2 id="doubledNum"></h2>
8       <h2 id="doubledString"></h2>
9       <script type="text/javascript" src="Test.dart.app.js"></script>
10    </body>
11  </html>
```

Si avranno così per risultato due funzionalità differenti che però riprendono le richieste iniziali dell'interfaccia. Proseguendo e ampliando un simile progetto si avrà in futuro una base molto solida e lineare, di facile comprensione e gestione, e non un susseguirsi via via più pesante di funzionalità, come per esempio si ha con javascript. In questo modo si comprendono, anche se attraverso un esempio molto semplice, le grandi potenzialità di un linguaggio come Dart che permette di strutturare il software in modo metodico e ingegneristico;

Tipizzazione opzionale

Di seguito un esempio di codice non tipizzato in Dart, che crea una nuova classe Space che ha i parametri x e y e due metodi: scale() e distance().

```
1
2  class Space {
3    var x, y;
4    Space(this.x, this.y);
5    scale(factor) => new Space(x*factor, y*factor);
6    distance() => Math.sqrt(x*x + y*y);
7  }
8
9  main() {
10   var a = new Space(2,3).scale(10);
11   print(a.distance());
12 }
```

Questo è come il codice diventa con l'aggiunta dei tipi che garantiscono che x, y e factor siano di tipo num e che Space contenga due valori di tipo num.

```
1
2  class Space {
3    num x, y;
4    Space(num this.x, num this.y);
5    Space scale(num factor) => new Space(x*factor, y*factor);
6    num distance() => Math.sqrt(x*x + y*y);
7  }
```

```
8  void main() {
9    Space a = new Space(2,3).scale(10);
10   print(a.distance());
11 }
```

Ad esempio, con x e y tipizzati un'eventuale creazione dell'oggetto Space con valori diversi dal tipo num creerebbe un eccezione grazie alla quale il programmatore potrebbe intervenire risolvendo velocemente il problema; in alternativa non si riscontrerebbe nulla se non nel momento di utilizzo di tali variabili a runtime.

Sunflower: HTML5, tipizzazionee e gestione Eventi

Analizzeremo ora un semplice esempio di interazione tra Dart e HTML5 per la creazione di un SunFlower: questo esempio utilizza **Canvas di HTML5** per il render di un'immagine che rappresenta il centro del fiore: Sunflower.
Il metodo migliore per la creazione di una UI statica come in questo caso, è di usare direttamente HTML. Ecco il codice HTML dell'esempio, il quale include l'heading text, l'oggetto Canvast, uno slider di HTML5 per l'input (slider), e una immagine che mostra l'equazione utilizzata per il render del sunflower :

```
1
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>Sunflower Demo</title>
6      <link type="text/css" rel="stylesheet" href="sunflower.css">
7    </head>
8    <body>
9      <h1>drfibonacci's Sunflower Spectacular</h1>
10
11     <p>A canvas 2D demo.</p>
12
13     <div id="container">
14       <canvas id="canvas" width="300" height="300" class="center"></canvas>
15       <form class="center">
16         <input id="slider" type="range" max="1000" value="500"/>
17       </form>
18       <br/>
19       <img src="math.png" width="350px" height="42px" class="center">
20     </div>
21
22     <footer>
```

Sunflower.png

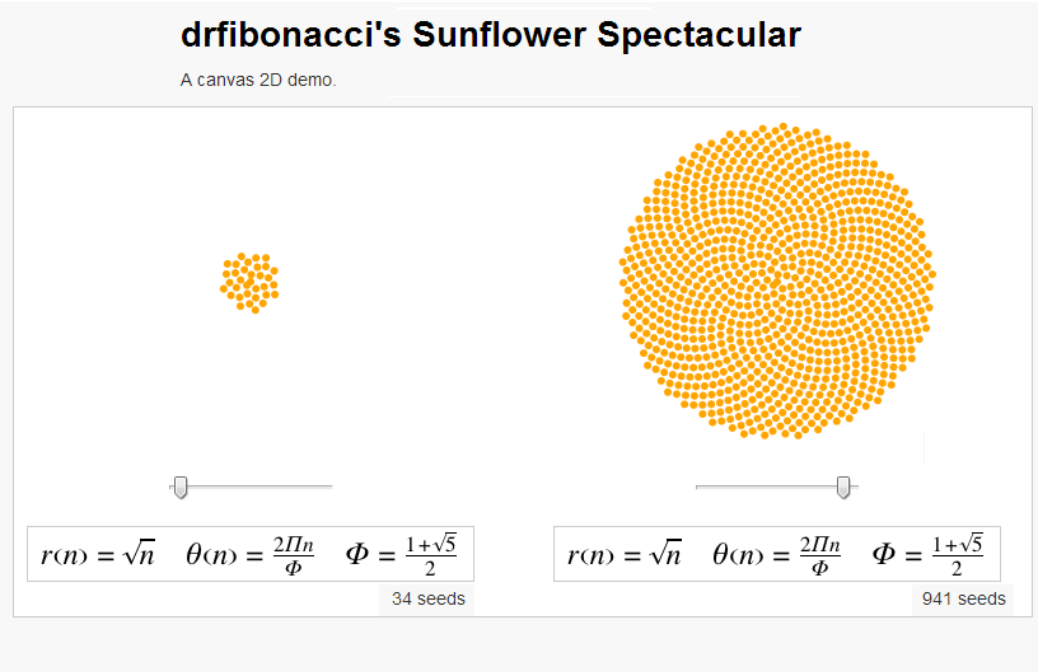

Figura 3.2: Fibonacci SunFlower

```
23          <p id="summary"> </p>
24          <p id="notes"> </p>
25        </footer>
26
27        <script type="application/dart" src="sunflower.dart"></script>
28        <script src="http://dart.googlecode.com/svn/branches/bleeding_edge/dart/client/dart.js"
29      </body>
30    </html>
```

Si nota dall'Html che il tag script "type" contiene il sorgente di tipo"application/dart": ciò permette di eseguire lo script direttamente da un browser che supporta Dart(al momento solamente Dartium), ma allo stesso tempo per tutti gli altri browser che non hanno la Dart VM, viene caricato il file "dart2js" per eseguire il programma in javascript. Si vuole sottolineare a questo punto che per assurdo quindi, si potrebbe utilizzare Dart anche solo come strumento di strutturazione del codice, esportandolo poi in javascript per l'esecuzione (anche se ovviamente non è questo l'obbiettivo di questo nuovo linguaggio di programmazione!). Come evidenziato nei paragrafi precedenti inoltre, sia che venga utilizzato il codice Dart, o che si usi il codice Javascript, l'esecuzione del programma partirà solamente

quando il DOM sarà pronto (su DOMContentLoaded).
Inoltre, a differenza di quanto si avrebbe avuto con javascript, nell'Html non compare traccia della **gestione degli eventi**, la quale è infatti controllata all'interno del codice Dart.
Analizzando velocemente il codice Dart del Sunfluwer si noterà che, come spiegato nei paragrafi introduttivi a Dart, si partirà con un main(): in questo caso viene inizzializzato l'oggetto Canvas, registrato un **handler per gli eventi**, e disegnato il primo frame.

```
1
2  // Copyright (c) 2012, the Dart project authors.  Please see the AUTHORS file
3  // for details. All rights reserved. Use of this source code is governed by a
4  // BSD-style license that can be found in the LICENSE file.
5
6  #library('sunflower');
7
8  #import('dart:html');
9  #import('dart:math', prefix: 'Math');
10
11 final SEED_RADIUS = 2;
12 final SCALE_FACTOR = 4;
13 final TAU = Math.PI * 2;
14
15 final MAX_D = 300;
16 final ORANGE = "orange";
17
18 num centerX, centerY;
19 int seeds = 0;
20
21 var PHI;
22
23 main() {
24   PHI = (Math.sqrt(5) + 1) / 2;
25
26   CanvasElement canvas = query("#canvas");
27   centerX = centerY = MAX_D / 2;
28   var context = canvas.context2d;
29
30   InputElement slider = query("#slider");
31   slider.on.change.add((Event e) {
32     seeds = Math.parseInt(slider.value);
33     drawFrame(context);
34   }, true);
35
36   seeds = Math.parseInt(slider.value);
37
38   drawFrame(context);
39 }
```

Vengono dichiarate molte costanti e variabili che sono tipizzate escplicitamente; ciò come visto è un enorme aiuto per il mantenimento del software. Inoltre l'utilizzo della variabile di tipo num potrebbe far pensare che essa sia una pimitiva in Dart(a causa dell'utilizzo di una notazione lowercase sulla prima lettera) cosa che non è possibile in quanto Dart non ha primitive: ciò infatti è dovuto all'utilizzo di librerie fondamentali(L'utilizzo di num elimina la necessità di converitre esplicitamente da int a double e viceversa in quanto vengono gestite come un unico tipo).
Infine uno sguardo al metodo che in definitiva disegna l'immagine :

```
1  /**
2   * Draw the complete figure for the current number of seeds.
3   */
4  void drawFrame(CanvasRenderingContext2D context) {
5    context.clearRect(0, 0, MAX_D, MAX_D);
6
7    for (var i = 0; i < seeds; i++) {
8      var theta = i * TAU / PHI;
9      var r = Math.sqrt(i) * SCALE_FACTOR;
10     var x = centerX + r * Math.cos(theta);
11     var y = centerY - r * Math.sin(theta);
12
13     drawSeed(context, x, y);
14   }
15
16   displaySeedCount(seeds);
17 }
18
19 /**
20  * Draw a small circle representing a seed centered at (x,y).
21  */
22 void drawSeed(CanvasRenderingContext2D context, num x, num y) {
23   context.beginPath();
24   context.lineWidth = 2;
25   context.fillStyle = ORANGE;
26   context.strokeStyle = ORANGE;
27   context.arc(x, y, SEED_RADIUS, 0, TAU, false);
28   context.fill();
29   context.closePath();
30   context.stroke();
31 }
32
33 void displaySeedCount(num seedCount) {
34   query("#notes").text = "${seedCount} seeds";
35 }
```

Il metodo drawFrame() compie tutti i calcoli mentre drawSeed() disenga sull'oggetto Canvas(in realtà sul CanvasRenderingContext2D).

Attraverso l'utilizzo combinato dell'**HTML5** e della **tipizzazione di Dart**, si possono creare interfacce molto complesse in maniera veloce e facile.

Capitolo 4

Conclusioni

Riprendendo l'introduzione di questa tesi, la rapidità alla quale tecnologia e web si stanno espandendo richiede un nuovo modo di affrontare la progettazione e lo sviluppo di software per la rete.
La necessità di dover affrontare progetti sempre più complessi porta ad una normale evoluzione verso un modello di programmazione strutturato. Dart si pone proprio come mezzo per affrontare questo cambiamento e risolvere le problematiche che riguardano il Web moderno; un linguaggio orientato agli oggetti di facile utilizzo per tutti coloro che si sono trovati sino ad oggi a sviluppare software locale in ambienti object-oriented e allo stesso tempo con funzionalità e potenzialità tali da poter creare applicazioni distribuite in maniera metodica e chiara; l'introduzione di un sistema basato agli oggetti appunto, garantisce le proprietà strutturali che sono richieste per sviluppare anche software complesso.
Inoltre grazie alla tipizzazione delle variabili si può fare affidamento su uno strumento più preciso, anche per la stessa analisi e correzione del software.
Nonostante però le forti caratteristiche che lo accomunano al javascript, Dart risulta un grosso cambiamento per chi ha per anni programmato con questo linguaggio, pur lasciando in ogni caso una grende libertà espressiva al programmatore.
In conclusione il progetto Dart è ancora in fase di sviluppo, ma ha delle buone potenzialità; si ricorda tuttavia,che per eseguire il codice Dart e per ottenere performance superiori all'attuale javascript i browser dovran-

no implementare la Dart Virtual Machine, un particolare che sicuramente ostacolerà molto le ambizioni di Google: non tutte le software house produttrici di browser, infatti, potrebbero essere disposte ad accettare tali imposizioni: Microsoft ad esempio non ha fatto segreto di non gradire tale linguaggio, e preferisce restare fedele a JavaScript per il quale sta lavorando ad un progetto parallelo chiamato TypeScript, che si pone gli stessi obiettivi di Dart: ampliare javascript attraverso l'introduzione di quelle stesse funzionalità viste in Dart, dalla tipizzazione opzionale, all'introduzione di classi e interfacce; Microsoft inoltre ha giustificato tali innovazioni come un naturale adattamento di javascript al futuro standard dell'ecmascript 6 che prevede appunto l'introduzione di calssi e interfacce ad esempio, e non come semplice modellazione ad immagine e somiglianza di altri suo linguaggi proprietari come c sharp, dal quale è molto evidente si è ispirata). Allo stato attuale non possiamo fare altro che tenere d'occhio l'evoluzione di questi progetti e vedere come il web risponderà a tali cambiamenti.

Bibliografia

[1] W3C
http://www.w3.org/

[2] DartLang
http://www.dartlang.org/

[3] Dartr.com
http://dartr.com/

[4] Chromium Blog
http://blog.chromium.org/

[5] Dartwatch
http://dartwatch.com/

[6] Techrepublic
http://www.techrepublic.com/blog/webmaster/getting-started-with-google-dart/

[7] Blog Sviluppare in Rete
http://sviluppare-in-rete.blogspot.it/

[8] Html5 - La Guida Italiana.
http://www.guidahtml5.com/

[9] HTML.it
http://www.html.it/

[10] Html5today
http://www.html5today.it/

[11] XHtml.it
http://xhtml.html.it/

[12] Sun
http://www.sun.com

[13] Wikipedia
it.wikipedia.org/

Ringraziamenti

Il primo ringraziamento va alla mia famiglia che mi ha aiutato, sostenuto e söpportato per tutti questi anni.
A Martina e Francesca che mi hanno sempre accompagnato in momenti felici e mi hanno sostenuto con la loro spontaneità ed amicizia e grazie alle quali ho trascorso mille pranzi in spensieratezza tra una lezione e l'altra, ammorbidendo le (a volte lunghe) giornate di lezione.
A Cornel che mi affianca in questo percorso dal primo giorno e che lentamente sto traviando verso il lato oscuro della forza!
..e a tutti gli altri ragazzi che ho incontrato in questo cammino e che spero di poter continuare a conoscere in futuro.
Infine, potrà sembrare inusuale e fuoriluogo, ma credo che un ultimo ringraziamento debba rivolgerlo a me stesso, per aver fatto le mie scelte e per averle portate avanti senza farmi influenzare da nessuno; per aver concluso questo percorso e per la voglia e l'intenzione di affrontarne altri mille.