Typescript: Javascript per applicazioni reali

#### di [Andrea Boschin](http://mvp.microsoft.com/profiles/Andrea.Boschin) – Microsoft MVP

* 1. 

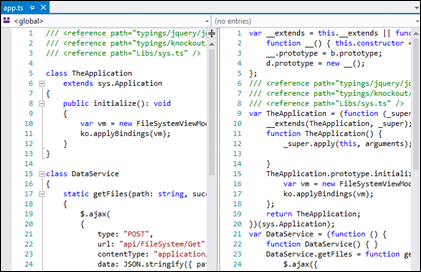
*Marzo, 2013*

Se mi guardo indietro mi vedo, numerosi anni fa, seduto sulla panchina del binario 3 della stazione di Venezia Mestre - probabilmente in attesa del treno che mi portava al lavoro - mentre leggo avidamente di un interessante linguaggio che *Netscape* aveva realizzato e *Microsoft* aveva introdotto in Internet Explorer 3.0; tale *Javascript*. La lettura, del tutto tecnica e didascalica ebbe la capacità di svegliare quelle attenzioni per le questioni "*Rich Internet*" che tutt'oggi mi accompagnano in varie esperienze.

Javascript, infatti, fu allora la risposta alle necessità che fin da subito permearono l'esperienza dello sviluppo di applicazioni web, in cui il gap nella user experience era talmente ampio rispetto quella della classiche applicazioni desktop da risultare indigesto ai più. Poter in qualche modo accedere alla dinamicità della pagina sul client e migliorare l'interazione con l'utente, era una necessità sentita e in qualche modo Javascript, in compagnia di DHTML, apriva un barlume di speranza.

Paradossalmente, oggi a distanza di almeno 17 anni da allora - in termini informatici almeno un paio di ere geologiche - Javascript è assurto al linguaggio per eccellenza, non esclusivamente dedicato allo sviluppo "rich" ma ormai con ampi spazi anche server-side. Dopo essere passati per numerose esperienze, che hanno visto alti e bassi, corsi e ricorsi, parziali abbandoni e ritorni, alla fine l'unico vero linguaggio che può vantare l'aggettivo "*cross-platform*" è Javascript.

Ma nonostante la longevità, anche nella sua più recente standardizzazione che va sotto il nome di *EcmaScript 5.0*, Javascript soffre dei problemi tipici dei linguaggi di scripting. In particolare la mancanza di tipi e la sua peculiare visione dell'object-orientation che omette concetti importanti quali l'incapsulamento e l'ereditarietà. A causa di questi problemi, nello sviluppo di applicazioni reali Javascript diventa un linguaggio ostico e, troppo spesso, pericoloso per la sua capacità di *digerire* qualunque cosa salvo poi *scoppiare* nel momento peggiore e cioè dopo il rilascio in produzione.

* 1. 

E' questa la ragione per cui molti si stanno orientando ad un nuovo tipo di strumenti, che abbia la capacità di tutelare lo sviluppatore mediante la *type-safety* e una programmazione *object oriented* vera, senza però perdere tutti gli indiscutibili vantaggi di Javascript con il cross-platform in testa a tutti. Microsoft in questo campo si sta muovendo rapidamente con la presentazione di un nuovo linguaggio denominato *Typescript*, giunto oggi alla versione *0.8.2*.

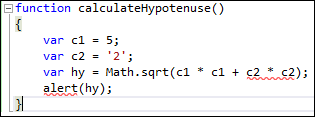
Pur trattandosi di una preview e omettendo ancora molti costrutti che uno sviluppatore normalmente si potrebbe aspettare, Typescript è un linguaggio sofisticato che ha la preziosa caratteristica di estendere Javascript, senza però richiedere un "interprete" nuovo nel browser, perchè il risultato della sua compilazione è Javascript. Un sorgente Javascript è a tutti gli effetti un sorgente Typescript perfettamente valido. Un sorgente Typescript genera comunque e sempre un sorgente Javascript valido. La parte del leone la fanno qui il compilatore "tsc.exe" e l'IDE di sviluppo, che anche se non necessariamente deve essere *Visual Studio 2012*, qualora lo si utiizzi è in grado di portare l'esperienza di sviluppo a livelli del tutto paragonabili a quelli che si hanno con linguaggi di alto livello come C#.

Ma andiamo con ordine: le prime prove con Typescript si possono già fare online nel [playground](http://www.typescriptlang.org/Playground/). Il compilatore Typescript infatti è scritto in Typescript - *per inciso è anche open source* - perciò è in grado di girare perfettamente all'interno del browser e in questa pagina è possibile scrivere, compilare e provare le prime righe di questo linguaggio, vedendone al contempo il risultato in termini di javascript prodotto. In alternative è possibile [scaricare](http://www.typescriptlang.org/#Download) i tool per Visual Studio e avere così una esperienza integrata, garantita da alcuni template e da un editor suddiviso in due aree che mostrano assieme sorgente e compilato.

Possiamo quindi sperimentare facilmente, e per farlo proviamo ad immettere il seguente codice nell'editor online oppure se preferite in Visual Studio 2012:

* 1. Typescript
  2. 1: function calculateHypotenuse()
  3. 2: {
  4. 3: var c1 = 5;
  5. 4: var c2 = '2';
  6. 5: var hy = Math.sqrt(c1 \* c1 + c2 \* c2);
  7. 6: alert(hy);
  8. 7: }

Ad un occhio attento appare evidente che, purtrattandosi di codice perfettamente legale per Javascript, esso poi genererà un errore di runtime dovuto al fatto che il valore di *c2* è impostato come una stringa invece che come un valore numerico, richiesto dalla funzione *Math.sqrt()*. Inoltre, un po' meno evidente ma anch'esso sorgente di un ulteriore errore, il valore *hy* ritornato è un numerico e non può essere passato direttamente alla funzione *alert()* che richiede una stringa. Il codice così com'è incollato nell'editor di Visual Studio ci metterà immediatamente in guardia proprio a causa dei suddetti problemi:

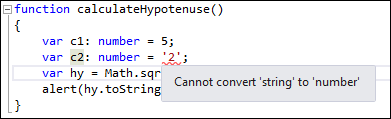
* 1. 

Visual Studio 2012, mediante le sottolineature in rosso evidenzia il problema, pur trattandosi di codice Javascript senza alcun costrutto particolare di Typescript.

Il compilatore Typescript infatti è in grado di *inferire* il tipo delle variabili dal loro utilizzo e di conseguenza segnala l'anomalia. Ovviamente, nella parte laterale, il codice Javascript non sarà generato ma sarà sostituito da un commento che riporta gli errori riscontrati.

Piuttosto che correggere semplicemente gli errori, lavorando con typescript, però possiamo *blindare* il codice dichiarandone i tipi.

Qui sotto vediamo il codice modificato e possiamo notare che a questo punto l'errore, visualizzato da Visual Studio, si è spostato nel punto, più opportuno, in cui avviene l'assegnazione. Il tooltip stesso evidenzia la causa dell'errore molto chiaramente. Poco sotto, parzialmente nascosto dal tooltip, viene usato toString() per fornire una stringa ad alert.

* 1. [](file:///C:\Users\BN\AppData\Local\Temp\WindowsLiveWriter1286139640\supfilesF8A97CD\image%5b11%5d.png)

La sintassi di Typescript è piuttosto semplice e *leggera* rispetto a javascript, lasciando per quanto possibile inalterato il codice originario. Il proposito del team è di mantenere per quanto possibile [una compatibilità](http://en.wikipedia.org/wiki/TypeScript#ECMAScript_6_support) con lo standard *Ecmascript 6.0* in corso di definizione, così da mantenere sempre inalterato il paradigma che sostiene che un sorgente Javascript è un Typescript valido. Una volta corretto l'ultimo errore nell'esempio avremo finalmente un output Javascript come segue:

* 1. Typescript
  2. 1: function calculateHypotenuse() {
  3. 2: var c1 = 5;
  4. 3: var c2 = 2;
  5. 4: var hy = Math.sqrt(c1 \* c1 + c2 \* c2);
  6. 5: alert(hy.toString());
  7. 6: }

Questo esempio banale, è però significativo e dà la misura di come, grazie al supporto di uno strumento di sviluppo quale il compilatore e in parte dell'IDE di Visual Studio 2012 in grado di sfruttarlo opportunamente, saremo in grado di scrivere codice di qualità che tuteli il nostro operato evitando che i normali errori si presentino nei momenti meno opportuni.

Typescript contiene una serie di altre feature molto importanti, come la definizione di interfacce e tipi custom, che lo arricchiscono di una espressività che spesso è limitata dai costrutti molto arzigogolati di Javascript. Spesso tali feature sono del tutto sostitutive e simulano a compile time feature che Javascript non include. E' il caso ad esempio dei metodi e delle proprietà private che, pur non essendo supportate da Javascript in alcun modo, in Typescript trovano applicazione e un eccellente supporto di intellisense in Visual Studio 2012. Il codice compilato naturalmente perderà questa caratteristica ma solo dopo che essa ha perso la sua utilità.

#### di [Andrea Boschin](http://mvp.microsoft.com/profiles/Andrea.Boschin) - Microsoft MVP

*twitter*: @aboschin

*blog italiano*: <http://blog.boschin.it>

*blog inglese*: <http://xamlplayground.org>

*facebook***:** <http://www.facebook.com/thelittlegrove>

*profilo***:** <http://slpg.org/AndreaBoschin>

Articolo pubblicato anche [sul Blog italiano](http://blog.boschin.it/post/2013/03/07/Typescript-Javascript-per-applicazioni-vere.aspx)