Funzionale alla Statale

Marco Faustinelli – sr. web seveloper, Onebip

Si parla tanto di JavaScript, spesso a sproposito

Obiettivi di questa esercitazione

- Presentare JavaScript come linguaggio per lo sviluppo di applicazioni web
- Mostrare l’utilizzo di JS come linguaggio per programmazione funzionale

JS è nato per operare nel browser, interagendo col DOM per modificare la grafica di una pagina HTML

Esempi di interazioni tra un programma JS e il DOM:

- DOM = modello a oggetti del documento (ossia della pagina web)
  - Generato dal motore di rendering del browser, interpretando codice HTML
- Tutte le chiamate al DOM sono disponibili all'interprete JS del browser
  - Creazione/modifica/distruzione elementi di pagina
  - Settaggio di event listeners per interagire con l'utente

JS è nato in maniera caotica e soffre da sempre a causa di scelte iniziali affrettate e/o mal coordinate.

Nonostante il nome che porta, JS è quanto di più lontano da Java si possa immaginare

Caratteristiche salienti di JavaScript:

- Linguaggio interpretato dinamicamente
- Singolo thread di esecuzione con coda per jobs ed eventi
- Funzioni come first-class objects
- Ereditarietà prototipale. Lexical scoping

Questi tratti di JS vengono da una scelta iniziale cosciente e senza compromessi; essi danno quindi una precisa identità e una ottima ragion d'essere al linguaggio.

...però bisogna conoscerli!

JavaScript è fondamentalmente un linguaggio non tipizzato

- I tipi di JS sono un concetto estremamente «volatile»:
  - Operatore **typeof** riconosce una mezza dozzina di tipi nativi differenti
  - Le variabili possono essere riassegnate a un tipo diverso
  - Molti operatori sono overloaded per operare su svariati tipi nativi
  - Casting dinamico, silenzioso ed implacabile

Freedom requires discipline

Voler incastrare a forza i tipi dentro JS è una futile complicazione

First-class functions e lexical scope consentono programmazione funzionale con naturalezza

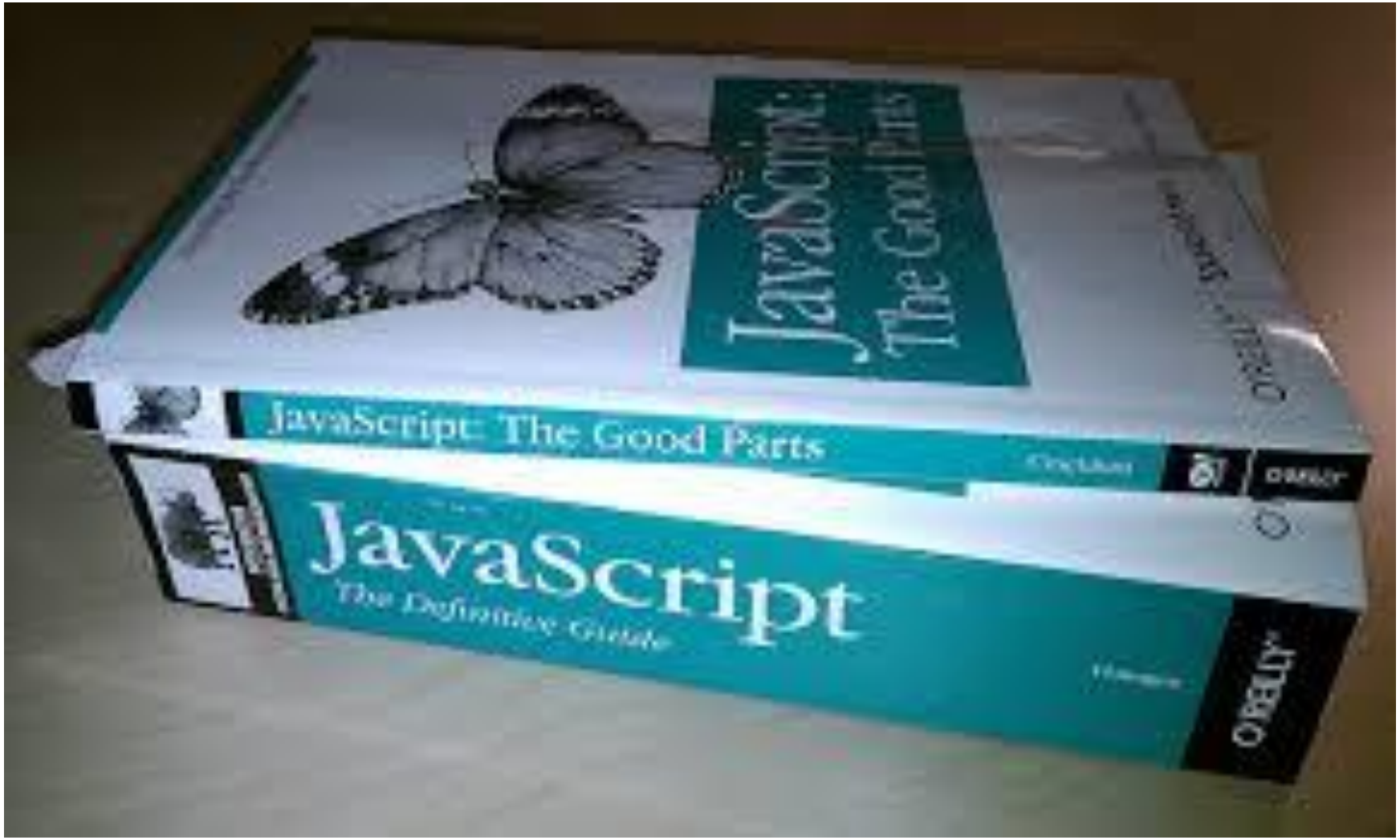
Dove il JS funzionale brilla:

- Named functions (sempre riconoscibili) al posto dei tipi sorvegliati dal compilatore
- Gestione capillare della visibilità e dell'accesso ai valori
- Buona dotazione di metodi puri
- Possibilità di pattern matching (functional style, però...)

Dove il JS funzionale richiede esperienza e prudenza:

- Immutabilità garantita solo nello scope delle funzioni
- Variabili globali sempre pronte a mordere
- No tail call optimisation (yet...)

E’ facile criticare JavaScript, ma preferiremmo forse avere BASIC nel nostro browser?



La diffusione e la robustezza di JS hanno favorito la proliferazione di linguaggi che compilano in codice JS

Due esempi dei tanti compilatori verso JavaScript:

- ClojureScript
- Khepri

Piccolo elenco di fonti

- ‘Javascript, the Good Parts’ (D. Crockford)
- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- underscorejs.org && ‘Functional Javascript’ (M. Fogus)
- Javascript Allongè (R. Braithwaite)
- Axel Rauschmayer, Matt Might, Matt Bierner