Tecnologie e Applicazioni dei Sistemi Distribuiti

Magistrale in Teoria e Tecnologia per la Comunicazione

Flavio De Paoli
flavio.depaoli@unimib.it
Giuseppe Vizzari
giuseppe.vizzari@unimib.it

●●● INSIDE&S Lab ●●●
http://inside.disco.unimib.it/

Programmazione in JavaScript

Parte I

●●● INSIDE&S Lab ●●●
http://inside.disco.unimib.it/

Code writing

- Code is a form of communication between the people who write and maintain it and is only incidentally executable on a machine, which we call a computer.
- Communication is just a fancy word for storytelling, something that humans have probably been doing since before we acquired language.
- Unless you are an accomplished surrealist, you tell a story by starting at the beginning, then over the course of time expose the reader to more of the details, finally arriving at the end where, hopefully, the reader experiences a satisfying bit of closure.
- The goal of the writer (or coder) is to form in the mind of the reader the same image the writer had.
- That is the process of communication, and it doesn't matter if it's prose, program or poetry—at the end of the day, if the recipient of our message has no clue what we meant, then all was for naught.

George V. Neville-Neil. 2018. The obscene coupling known as spaghetti code.
Commun. ACM 61, 10 (September 2018), 27-28. DOI: https://doi.org/10.1145/3267356

Programmazione imperativa

- Programmare significa definire una (o più) sequenze di azioni (dette algoritmi) che determinano la soluzione ad un problema dato.
- Un programma è la codifica delle azioni in un linguaggio comprensibile da un computer.
- Un programma, riceve dei dati in ingresso e produce dei dati in uscita che rappresentano la soluzione.
- Un linguaggio di programmazione definisce delle istruzioni (gli elementi base) che composti definiscono i diversi programmi.

Dispensa: «Introduzione alla programmazione» sul sito di elearning
`https://elearning.unimib.it/mod/resource/view.php?id=761558`

Programmazione imperativa

- Istruzioni (azioni) fondamentali:
  - Dichiarazione di variabili
    (dare un nome a spazi in memoria per contenere i dati)
    - Variabili semplici (che contengono un solo valore)
    - Variabili composte (che contengono più valori)
  - Assegnamento (scrivere un valore in una variabile)
  - Scelta condizionale (definire quale eseguire tra più sequenze di istruzioni)
  - Ripetizione (applicare una sequenza di azioni più volte su dati diversi)

  - Espressione (operazioni di trasformazione dei dati – come in matematica)
  - Funzione (assegnare un nome ad una sequenza di istruzioni)
  - Lettura/Scrittura (ricevere i dati da cui partire in ingresso e produrre le soluzioni in uscita)

JavaScript

- JavaScript è un linguaggio di scripting interpretato da un *engine*.
  In pratica viene eseguito dal browser.
  Nodejs permette l'esecuzione sul server, ma lo vedremo più avanti.
- Javascript permette di rendere le pagine html dinamiche, cioè di inserire dei programmi che modificano il comportamento e le visualizzazioni.
- Javascript è importante perché ha
  - la capacità di effettuare richieste in formato HTTP al server, in maniera trasparente all'utente
  - la funzione di rendere asincrona la comunicazione tra browser e web server
- Javascript può richiedere dati in formato testo puro e XML
  - Attualmente il formato più diffuso è JSON
- Tutorial disponibili online:
  - http://www.w3schools.com/js/default.asp (JavaScript tutorial)
  - http://www.w3schools.com/jsref/default.asp (JavaScript and HTML DOM Reference)

Elementi di programmazione (1)

Gli elementi fondamentali della *programmazione imperativa* sono

- ***Dichiarazione*** di variabili semplici: servono per memorizzare valori di un certo ***tipo***
  I tipi elementari sono: numeri (interi e decimali), caratteri, valori logici (true e false)

```
var age; (in JavaScript il tipo non è specificato)
var drink_alcohol;
```

- Istruzione di ***assegnamento***: assegna un valore ad una variabile

```
var name;
var name = value;
name = value;
```

```
1. age = 35;
2. age = anno_corrente – anno_nascita; (assegno risultato espressione)
3. drink_alcohol = false; (valori booleani true / false)
```

- Istruzione di ***scelta condizionale***: definisce due percorsi alternativi.

```
4. if (age < 18) {
5.     drink_alcohol = false;
6.     } else {
7.     drink_alcohol = true;
8. }
```

```
1. if (conditional expression) {
2.       istructions if true
3.    } else {
4.       istructions if false
5. }
```

NOTA: la parte `else` è opzionale

Per provare ad eseguire: https://eloquentjavascript.net/code/

Esercizio 1

Questo è un frammento di programma

```
1.  var age = 35;
2.  var drink_alcohol = false; // valori booleani true e false
3.  if (age < 18) { // età minore di 18
4.    drink_alcohol = false;
5.    } else {
6.    drink_alcohol = true;
7.  }
```

	Var	Value
1	age	35
2	drink_alcohol	false
3		false
4		
5		
6	drink_alcohol	true
7		

- È possibile migliorare il programma? (meno istruzioni: + efficienza – errori)

```
3.  if (age >= 18) {
4.    drink_alcohol = true;
5.    } else {
6.    drink_alcohol = false;
7.  }
```

Posso migliorarlo?

```
3.  if (age >= 18) {
4.    drink_alcohol = true;
5.    } // else non serve perché è già false
```

- Modificarlo cambiando la condizione in «età maggiore o uguale a 18»

Eseguire i primi programmi

Code Sandbox
Eloquent JavaScript

Link alla sandbox: https://eloquentjavascript.net/code/

You can use this page to download source code and solutions to exercises for the book Eloquent JavaScript, and to directly run code in the context of chapters from that book, either to solve exercises to simply play around.

Chapter: 0. Introduction | This chapter has no exercises | run code

```
1  var age = 35;
2  var drink_alcohol = false; // valori booleani true e false
3  if (age < 18) { // età minore di 18
4    drink_alcohol = false;
5    } else {
6    drink_alcohol = true;
7  }
8
9  console.log(age);
10 console.log(drink_alcohol);
```

```
35
true
```

L'istruzione **console.log(** *nome_variabile* **)** permette di visualizzare il valore della variabile indicata.

NOTA: dettagli la prossima lezione.

Elementi di programmazione (2)

- Un'***espressione*** è una combinazione di valori, variabili ed operatori che rappresentano un nuovo valore.
- JavaScript prevede operatori *unari*, *binari* e *ternari* a seconda che possano combinare rispettivamente uno, due o tre valori.
- ***Operatori aritmentici***: +, –, *, /, %(modulo o resto), combinati secondo le regole di precedenza matematiche, con uso di parentesi *tonde* per modificarle

```
9 % 4 + 5                genera il valore 6
4 + 5 * 6 + 7            genera il valore 41
(4 + 5) * (6 + 7)        genera il valore 117
```

9 / 4 + 5 genera il valore ?

- ***Operatori relazionali*** (o di *confronto*): >, >=, <, <=, === (uguale), !== (diverso)

```
4 > 2                    genera il valore true
5 != 4 + 1               genera il valore false
```

- ***Operatori logici***: && (and), || (or), ! (not)

```
5 > 2 && 3 !== 4         genera il valore true
true || 4 >= 6           genera il valore true
!5===5                   genera il valore false
```

NOTA: meglio usare le parentesi per evitare possibili errori.

```
(5 > 2) && (3 !== 4)   // true
true || (4 >== 6)      // true
!(5===5)               // false
```

https://www.html.it/pag/15220/operatori1/

Elementi di programmazione (3)

JavaScript Keywords

- Le istruzioni JavaScript spesso iniziano con una ***keyword*** (parola riservata) per indentificare l'azione che dovrà essere eseguita.
- Le parole riservate non possono essere usate come nome di variabili
- Alcune keywords:

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

Esercizi da svolgere

- Modificare l'esercizio 1 aggiungendo
  - Dichiarazioni di `anno_corrente` e `anno_nascita`
  - Assegnazione dei valori a queste variabili
  - Calcolo l'età come differenza
- Scrivere il frammento di programma che calcoli il perimetro di un rettangolo.
  - Dichiarare le variabili lato1, lato2, perimetro
  - Assegnare dei valori ai lati e assegnare al perimetro il risultato della espressione di calcolo
- Scrivere il frammento di programma che verifichi se, date le misure dei lati, un quadrilatero è un parallelogramma.

Elementi di programmazione (4)

- Variabili composte ***array***: servono per memorizzare sequenze di valori di un certo ***tipo***

```
1. var names = ["Anna", "Brittany", "Cinderella", "Diana"];
2. var ages = [15, 35, 28, 18];
```

- Accesso agli elementi degli array con indici da 0 a length-1 (length è la lunghezza array)

```
ages[2] = 28; // assegna 28 al terzo elemento dell'array
names[0] == "Anna" // espressione di confronto ed ha esito true
```

- Istruzione di ***ripetizione*** (o ***ciclo***): esecuzione ripetuta di istruzioni sotto condizione

```
3. var drink_alcohol = [ ];
4. for (i = 0; i < ages.length; i++) {
5.   if (ages[i] < 18) {
6.     drink_alcohol[i] = false;
7.   } else {
8.     drink_alcohol[i] = true;
9.   }
10.}
```

```
1. for (first_step; condition; step) {
2.   if (condition) {
3.     istructions if true
4.   } else {
5.     istructions if false
6.   }
7. }
```

Elementi di programmazione (4)

- Variabili composte ***array***: servono per memorizzare sequenze di valori di un certo ***tipo***

```
1. var names = ["Anna", "Brittany", "Cinderella", "Diana"];
2. var ages = [15, 35, 28, 18];
```

- Accesso agli elementi degli array con indici da 0 a length-1 (length è la lunghezza array)

```
ages[2] = 28; // assegna 28 al terzo elemento dell'array
names[0] == "Anna" // espressione di confronto ed ha esito true
```

- Istruzione di ***ripetizione*** (o ***ciclo***): esecuzione ripetuta di istruzioni sotto condizione

```
3. var drink_alcohol = [ ];
4. for (i = 0; i < ages.length; i++) {
5.   if (ages[i] < 18) {
6.     drink_alcohol[i] = false;
7.   } else {
8.     drink_alcohol[i] = true;
9.   }
10.}
```

```
1. for (first_step; condition; step) {
2.   if (condition) {
3.     istructions if true
4.   } else {
5.     istructions if false
6.   }
7. }
```

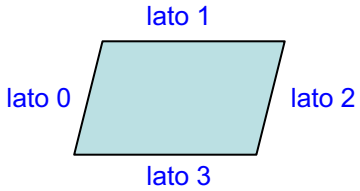
Che valore resta a `drink_alcohol` dopo l'uscita dal ciclo for? Come possiamo ricordare se `ages[0]` può bere? Provarlo con https://eloquentjavascript.net/code/ e visualizzare nome e rislutato per l'utente [0]

Esercizio 2

- Scrivere il frammento di programma che verifichi se, date le misure dei lati, un quadrilatero è un parallelogramma.
- Criterio: se i lati opposti sono uguali, allora il quadrilatero è un parallelogramma.

Svolgimento

- Ipotesi: conosco le misure dei 4 lati



- Possibile algoritmo:
  1. Controllo 0: Considero un lato e verifico se esiste un altro lato uguale
  2. Se esiste un altro lato uguale, allora faccio altri controlli, altrimenti posso concludere che non è un parallelogramma
  3. Controllo 1: se i lati uguali sono adiacenti e anche gli altri due sono uguali, allora posso concludere che è un parallelogramma (4 lati uguali)
  4. Controllo 2: se i lati uguali non sono adiacenti e gli altri due sono uguali tra loro, allora posso concludere che è un parallelogramma (lati opposti uguali)

Esercizio 2

- Scrivere il frammento di programma che verifichi se, date le misure dei lati, un quadrilatero è un parallelogramma.
- Criterio: se i lati opposti sono uguali, allora il quadrilatero è un parallelogramma.

```
1. var lati = [15, 35, 15, 88]; // ipotesi: ho i 4 valori delle lunghezze dei lati
2. var parallelogramma = false; // ipotesi: non è un parallelogramma
3. // Controllo 0: Considero un lato e verifico se esiste un altro lato uguale
4. var posizione = 0;
5. for (i = 1; i < lati.length; i++) {
6.   if (lati[0] == lati[i]) {
7.     posizione = i; // la posizione del lato uguale al lato 0, se esiste
8.   }
9. }
10.
11.// se la posizione trovata è diversa dalla posizione 0,
12.// allora potrebbe essere un parallelogramma
13.// serve confrontare gli altri due lati di posizione diversa da 0 e quella trovata
14.// ... completare ...
```