A black background with white text

Description automatically generated

Per "programmazione orientata agli oggetti"

(object-oriented programming) si intende un pattern di programmazione che si sviluppa attorno alla creazione

di oggetti software, composti di determinate proprietà

e abilità, e alla loro interazione reciproca.

Oggi risulta essere una delle tecniche di

programmazione più diffusa, soprattutto per applicativi

di medie-larghe dimensioni in cui è necessario dividere

agevolmente il lavoro tra diversi sviluppatori.

La metodologia OOP è colonna portante di alcuni linguaggi di programmazione (come Java o C#), ma risulta applicabile anche in altri linguaggi "ibridi" (come JavaScript).

JavaScript non nasce come un "classico" esempio di linguaggio di programmazione orientato agli oggetti; questo è dovuto soprattutto al fatto che al contrario

di altri linguaggi che utilizzano dei meccanismi di ereditarietà basati sulle classi (come Java, o C#), JavaScript utilizza un sistema prototipale per la definizione delle proprie logiche interne.

JavaScript tuttavia può venire utilizzato per la

scrittura di codice orientato agli oggetti, in quanto è possibile definire utilizzando la sua sintassi nativa le tre logiche principali dei linguaggi OOP:

- Incapsulamento

- Ereditarietà

- Polimorfismo

A screenshot of a computer

Description automatically generated

A differenza degli altri linguaggi presi in esempio, in JS

è possibile creare un oggetto direttamente, senza aver

creato precedentemente una classe, utilizzando la

cosiddetta "notazione letterale".

Un oggetto può contenere degli attributi, ovvero delle variabili/costanti che ne specificano le caratteristiche, e dei metodi, che ne specificano le abilità e le

operazioni che può compiere.

**Per accedere ai valori degli attributi** o per invocare i metodi, è necessario esplorare l'oggetto tramite l'utilizzo **dell'operatore ""** o specificare il nome della proprietà tramite parentesi quadre. A screenshot of a computer

Description automatically generated

Quando tuttavia è necessario creare molteplici oggetti con la medesima struttura, la notazione letterale risulta un impedimento, in quanto costringe a copiare/incollare l'elenco delle proprietà a mano ogni volta.

Come procedere se è necessario creare "in serie" numerosi oggetti tutti dotati delle stesse proprietà?

Al fine di creare oggetti dotati della stessa

struttura in modo sistematico, è possibile utilizzare

un costruttore. A screenshot of a computer code

Description automatically generated

**Un costruttore** ***è una speciale funzione*** (per

convenzione la prima lettera va specificata

maiuscola) che permette di creare, tramite la sua

invocazione con la keyword "new", un oggetto

"precostituito" di determinati attributi/metodi.

In un colpo solo vengono assegnate molteplici

coppie chiave/valore, che possono essere lette o sovrascritte.

I **metodi** descritti nel **costruttore** possono

contenere la keyword "this" al fine di accedere

alle proprietà assegnate all'oggetto creato.

Ma un costruttore può anche spingersi oltre:

durante la sua **invocazione tramite "new"** è

possibile passare alla funzione anche dei

parametri, che possono essere utilizzati dal

costruttore stesso in modo da inizializzare

determinate proprietà dell'oggetto appena creato.

Questo permette di ottenere un oggetto non più totalmente vuoto, ma con già dei valori assegnati agli attributi/metodi di nostra scelta.

L'implementazione e l'assegnazione delle

proprietà vengono descritte una volta sola

all'interno di NamedPerson, la funzione

costruttore. A screenshot of a computer

Description automatically generated

In JavaScript, la parola chiave **"this"** ha un

comportamento diverso da quello di molti altri linguaggi di programmazione. Essa **può essere usata in qualsiasi funzione**, anche se non si tratta del metodo di un oggetto.

Il valore di **"this"** viene **valutato al momento**

dell'esecuzione della funzione.

Per comprendere il valore di "this", la regola è

semplice: per invocare un metodo "func" di un oggetto "obj", si utilizzerà obj.func(); durante la

chiamata di func(), "this" si riferisce a obj, ovvero all'oggetto posto prima della notazione di accesso (in questo caso, il punto).

Il concetto di valutare "this" durante l'esecuzione della funzione ha i suoi pregi e difetti.

Un pregio ad esempio è la riutilizzabilità del

codice, in quanto è possibile riutilizzare la stessa funzione per oggetti diversi.

D'altro canto, come spesso accade in JavaScript, tutta questa flessibilità spesso rappresenta anche una debolezza e può essere fonte di errori.

Come abbiamo già visto, il concetto di "classe" in JavaScript non è alla base del linguaggio come in altri casi (Java, C# etc.), in quanto JS utilizza un meccanismo di ereditarietà basato sui prototipi.

Nelle ultime revisioni di JavaScript tuttavia, al fine di uniformare la sintassi e attirare nuovi developer senza doverli far rinunciare ai costrutti a loro familiari, è stata introdotta la keyword "class".

La **keyword "class"** non è altro che un metodo

alternativo **per la creazione di un costruttore**.

Gli oggetti creati a partire da una classe vengono definiti "istanze" della classe.

A screenshot of a computer code

Description automatically generatedÈ importante sottolineare che la keyword "class"

non aggiunge nulla alle funzionalità di JavaScript

e non ne altera assolutamente il modello di

ereditarietà; le fondamenta del codice, anche con questa aggiunta, rimangono le medesime.

**Di fatto il costrutto "class" svolge la stessa**

**funzione di un costruttore**, solamente con un'altra sintassi al fine di uniformare JavaScript dal punto di vista stilistico ad altri linguaggi.

**Possiamo definire il costrutto "class"** come uno sugar coating sulla già esistente funzione

costruttore.

A screenshot of a computer

Description automatically generated

Una funzionalità disponibile **solamente**

**utilizzando il costrutto "class"** tuttavia c'è: si

tratta della possibilità di creare una nuova

classe estendendo una già esistente,

utilizzando tramite la keyword "**extends".**

Utilizzando "extends" si crea **una sottoclasse** a

partire dall'originale, che erediterà tutti gli

attributi/metodi di quella iniziale permettendo di aggiungerne di nuovi.

E possibile dunque invocare dalla classe estesa anche i metodi/attributi della classe originale. A screenshot of a computer program

Description automatically generated

Se definiamo **una sottoclasse a partire da**

**un'altra**, molto probabilmente all'interno del

suo costruttore sarà necessario come prima

cosa invocare il costruttore della classe da cui

stiamo estendendo. Per farlo è sufficiente

invocare **"super()".**

**"super()" è un riferimento alla funzione**

**costruttore della classe originale.**

Nell'esempio a fianco, il costruttore della

classe Developer riceve tre parametri: due

saranno passati al costruttore della classe

Person per la sua inizializzazione, mentre

yearsOfExp viene assegnato ad un attributo

presente solamente negli oggetti di classe

Developer. A screenshot of a computer code

Description automatically generated"**super", più genericamente, rappresenta un**

**riferimento alla classe che stiamo**

**estendendo.**

Tramite esso è possibile anche invocare nella

classe estesa metodi e proprietà di quella

originaria, come indicato nell'esempio a

fianco.

Il metodo "meows" della classe Cat altro non

fa che invocare il metodo "roars" della classe

Lion.

A screenshot of a computer code

Description automatically generated

La **keyword "static"** definisce un **metodo**

**statico o un attributo statico per una classe**.

I membri statici possono venire **richiamati**

**solamente sulla classe stessa**, e non verranno

trovati all'interno delle istanze generate a

partire dalla classe.

I metodi **statici** vengono spesso **usati per**

**creare funzioni di utilità**, spesso atte alla

creazione/clonazione di oggetti, mentre le

proprietà statiche sono **utili per cache**,

**configurazione o per la definizione** di

qualsiasi altro dato non sia necessario

replicare nelle istanze. A screenshot of a computer program

Description automatically generated