Rimini 06/03/2023

AdriasOnline
Connessi per accogliere

Agenda

Gli Oggetti: i metodi e la parola chiave this
Gli Oggetti definiti con il costruttore

metodi degli oggetti

Come abbiamo già detto gli oggetti sono utilizzati per **rappresentare dei dati** (entità) del mondo reale come *Utenti*, *Giocatori*, *Ordini* e etc.. queste entità possono effettuare delle **azioni**. Ad esempio un Utente può effettuare un acquisto, commentare un post e etc.. **Queste azioni in Javascript vengono chiamate metodi**. Un metodo è rappresentato da una **funzione**.

```
var player = {nome: "Francesco", cognome: "Totti"};
player.esulta = function(){
    console.log("Gooooal!");
};
player.esulta(); //Viene stampato sulla console: Goooo
```

AdriasOn
Connessi per accog

I metodi degli oggetti

```
var player = {nome: "Francesco", cognome: "Totti"};
var esulta = function(){
    console.log("Gooooal!");
};
player.esulta = esulta;
player.esulta(); //Viene stampato sulla console: Goooo
```

AdriasOn
Connessi per accog

I metodi degli oggetti

```
player = {
   esulta: function() {
       console.log("Gooooal!");
   }
};

// la sintassi più breve
player = {
   esulta() { // equivalente a "esulta: function(){...}"
       console.log("Gooooal!");
   }
};
```

AdriasOn
Connessi per accog

I metodi degli oggetti

Esiste una sintassi più breve per i metodi in un oggetto letterale

this

E’ molto comune che, per eseguire determinate azioni, un metodo abbia necessità di **accedere alle proprietà** dell’oggetto.

Ad esempio, il codice dentro **player.esulta()** potrebbe aver bisogno del nome del player.

Per accedere all’oggetto un metodo può utilizzare la parola chiave **this**.

```
var player = {
  nome: "Francesco",
  esulta() {
      console.log(this.nome +": Gooooal!" ); // Francesco: Goooo
  }
};
```

AdriasOn
Connessi per accog

this

```
//Questo codice non produce nessun errore.
function esulta() {
  console.log( this.nome );
}
```

AdriasOn
Connessi per accog

this

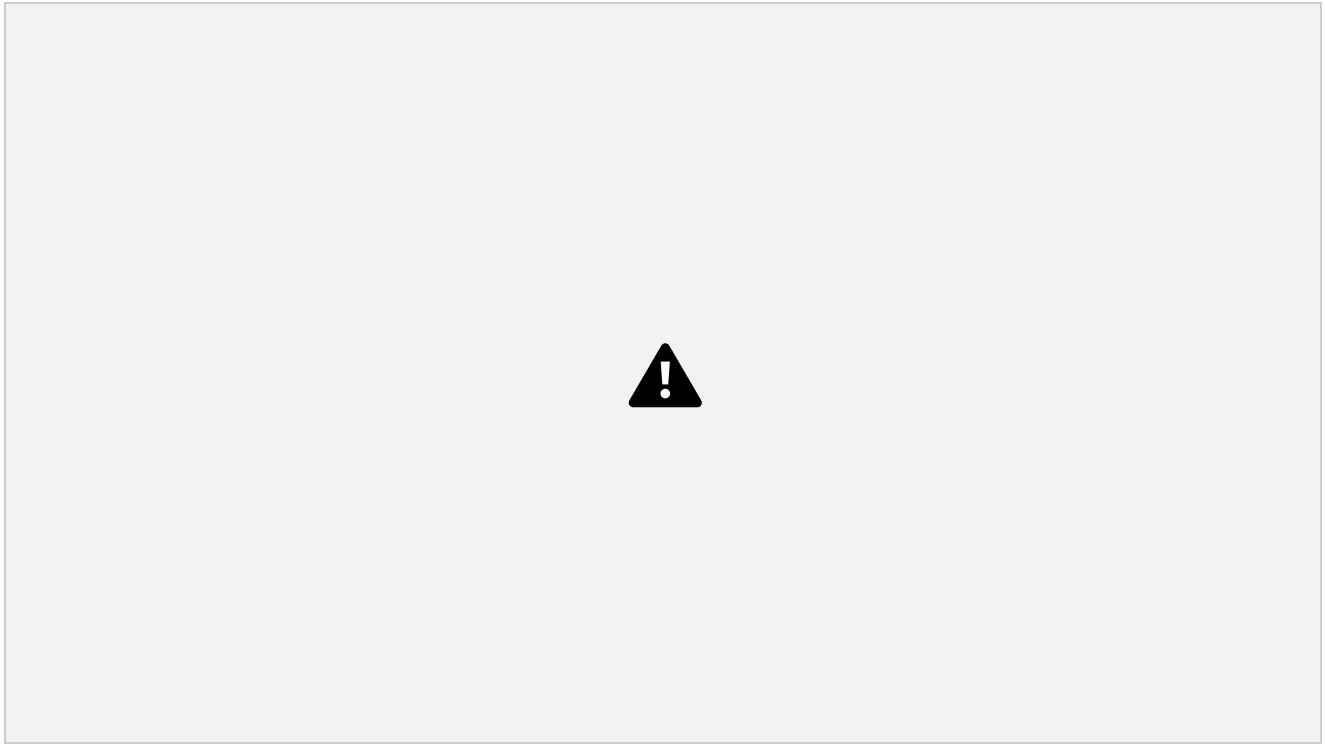
In JavaScript, la parola chiave **“this”** si comporta diversamente da come fa in molti altri linguaggi di programmazione. Essa può essere usata **in qualsiasi funzione**, anche se non si tratta del **metodo di un oggetto**.

```
var player = {nome: "Francesco"}
var professore = {nome: "Giuseppe"}
var fn = function(){console.log(this.nome);};
player.f = fn;
professore.f = fn;
player.f(); //Francesco
professore.f(); //Giuseppe
professore["f"](); //Giuseppe
```

AdriasOn
Connessi per accog

this

Il “this” viene interpretato dal motore Javascript in modo implicito a seconda dei casi. Ad esempio nel codice sottostante ad ogni invocazione del metodo f, il this accede alle proprietà degli oggetti corrispondenti.



Domande?

Esercitazione 1

Scrivi un programma per la gestione di un garage.

Definisci un oggetto che rappresenti un automobile, dovrà contenere almeno marca del veicolo e nome del modello.

Definisci un oggetto che rappresenti il garage.

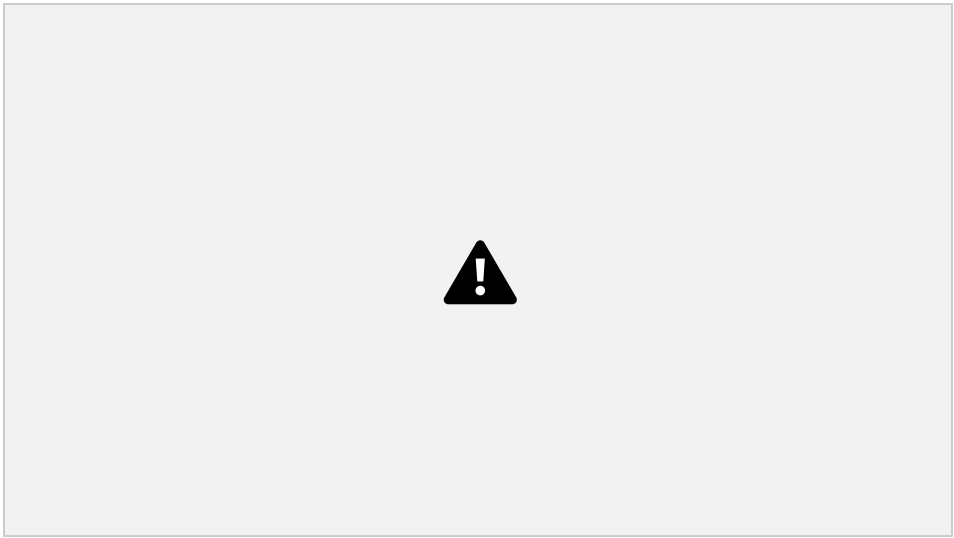
Scrivi una funzione che inserisca un veicolo nel garage.

Scrivi una funzione che prenda in input una marca e stampi i modelli presenti nel garage di quella stessa marca.

Scrivi una funzione che rimuove un veicolo dal garage.

Puoi usare un array come base per salvare le automobili.

Per togliere un elemento dall'array puoi usare il metodo arr.splice(i, 1); dove i è l'indice posizionale e 1 rappresenta il numero di elementi da eliminare.



Costruttore

Riepilogo della lezione sugli Oggetti

Gli oggetti sono **arrays associativi** con diverse caratteristiche speciali:

Possono memorizzare proprietà (coppie di chiave-valore) in cui:

- Il nome della proprietà (chiave) deve essere composta da una stringa (che può contenere anche simboli) . ● I valori possono essere di qualsiasi tipo

Per accedere ad una proprietà possiamo utilizzare:

- La notazione puntata: obj.property.
- La notazione con parentesi quadre obj["property"]. Questa notazione consente di accettare chiavi dalle variabili, come obj[varWithKey].

Operatori specifici:

- Per cancellare una proprietà: delete obj.prop.
- Per controllare se un una proprietà con un certo nome esiste: "key" in obj.
- Per iterare un oggetto: for(let key in obj).

Riepilogo

Le funzioni che vengono memorizzate come proprietà di un oggetto vengono dette “metodi”.

- I metodi consentono agli oggetti di “agire”, **come object.doSomething(). ●**
  I metodi possono riferirsi all’oggetto tramite **this**.
- Il valore **this** viene definito durante l’esecuzione (run-time).

Quando una funzione viene dichiarata, può utilizzare **this**, ma questo **this** non avrà alcun valore fino a che la funzione non verrà chiamata.

- Una funzione può essere copiata in vari oggetti.
- Quando una funzione viene chiamata come “metodo”: object.method(), il valore di this durante la chiamata si riferisce a object.

L’operatore new

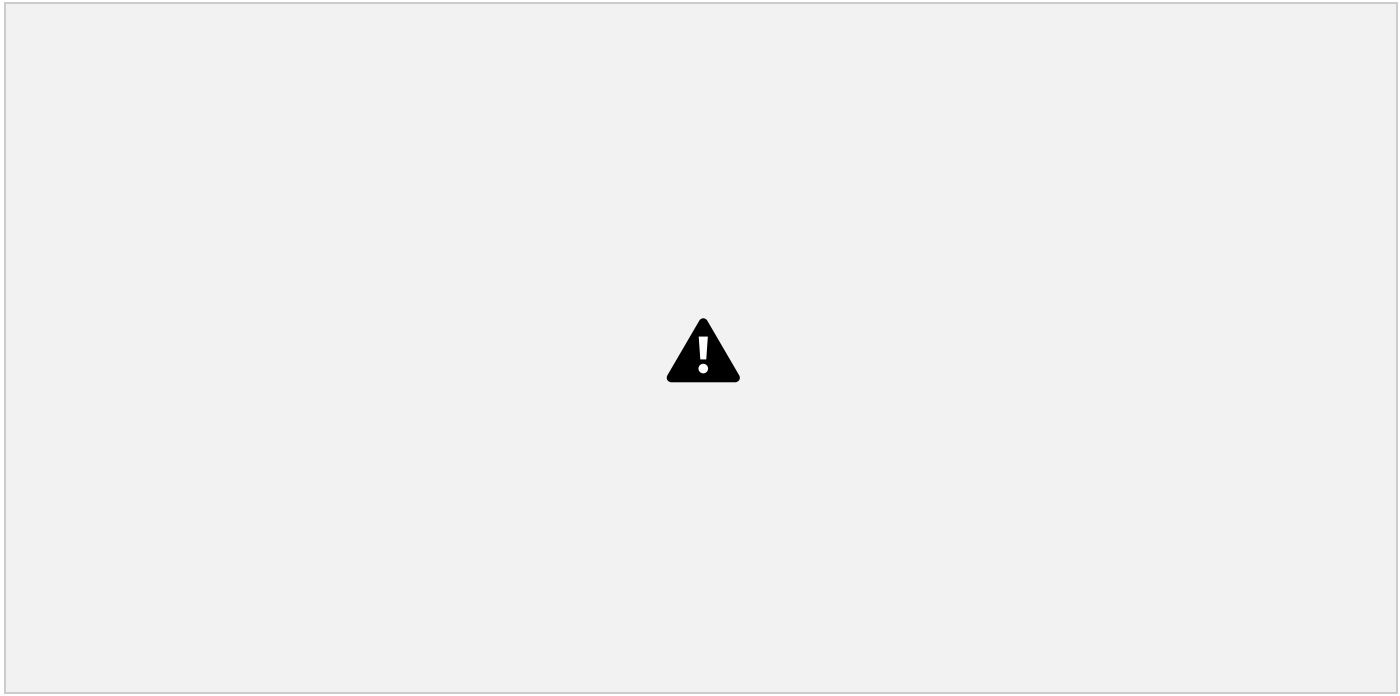
La sintassi **{...}** ci consente di creare un oggetto in modo descrittivo, però nella *programmazione orientata agli oggetti* spesso abbiamo il bisogno di **creare istanze di oggetti simili**, come ad esempio oggetti che rappresentano *gli utenti, gli ordini o le fatture*.

Questo può essere fatto utilizzando un costruttore e l'operatore **"new"**.

Costruttore

Tecnicamente un costruttore è una “normale” funzione, però ci sono due convenzioni:

- Vengono **denominati** con la **prima lettera maiuscola**. ● Questi d**ovrebbero essere eseguiti** solo con l’operatore **"new"**.

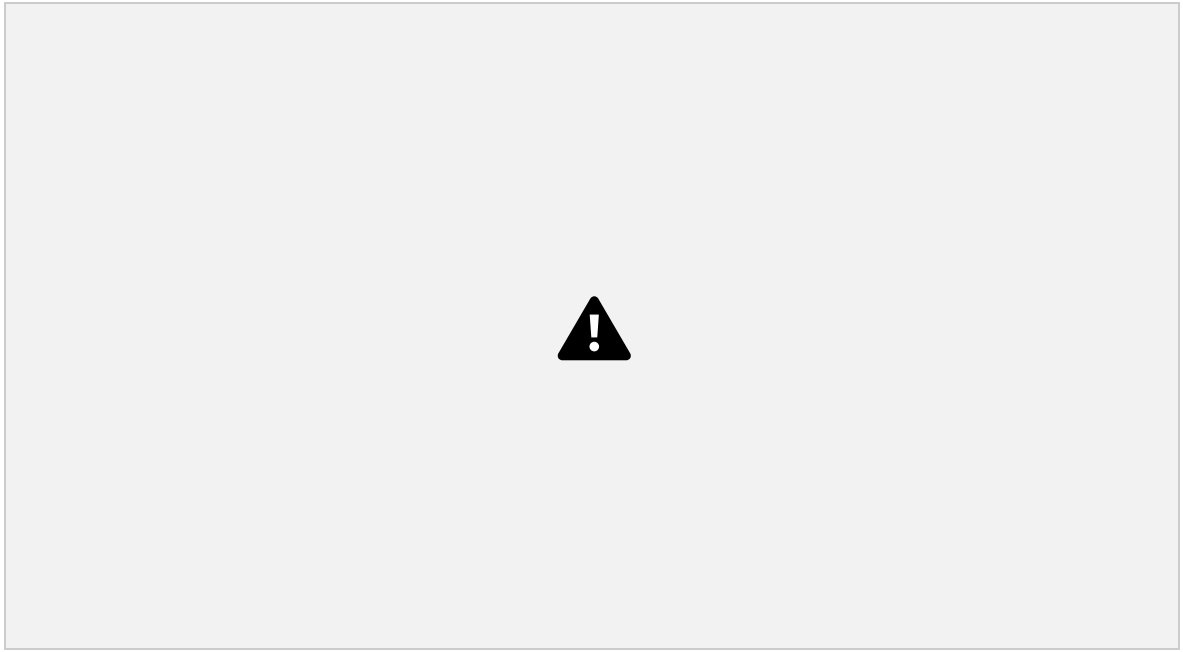


Costruttore

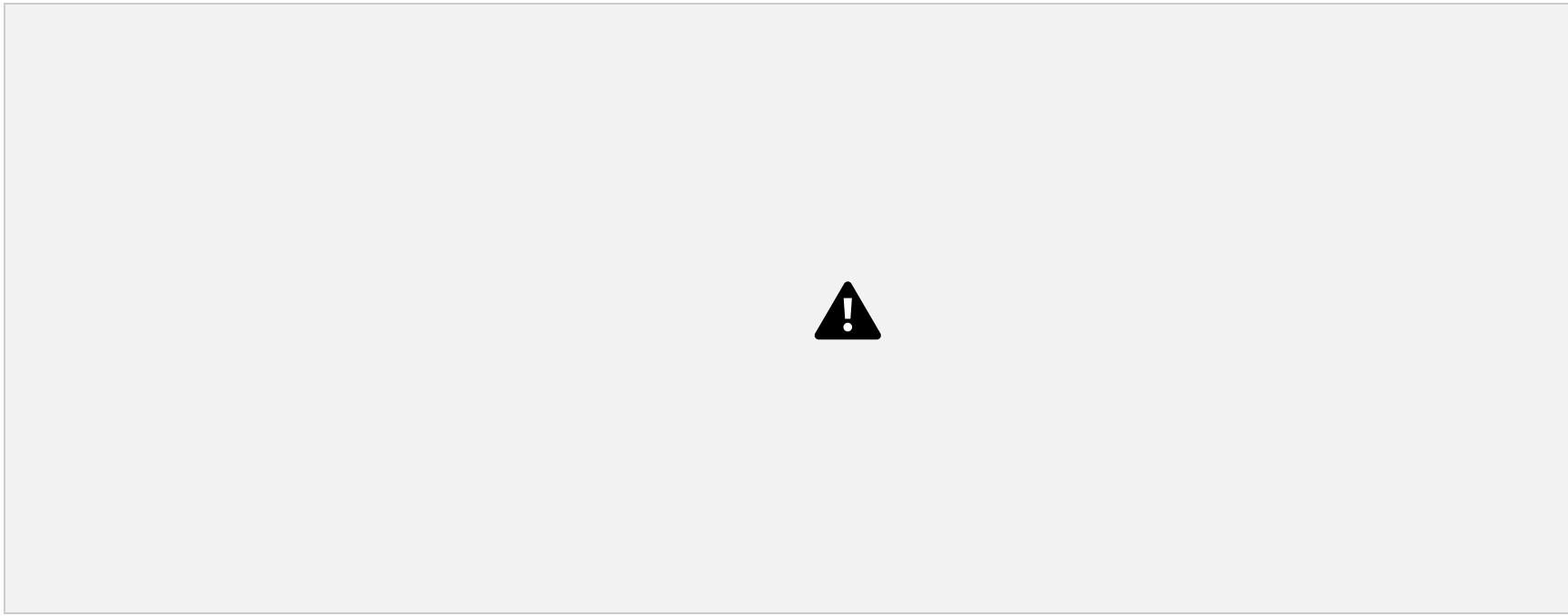
Costruttore

Quando una funzione viene eseguita con **new**, l'interprete Javascript esegue questi passaggi:

1. Un **nuovo oggetto**, **vuoto**, viene creato ed assegnato a **this**. 2. Viene eseguito il **corpo della funzione**. "Solitamente" questo modifica **this**, aggiungendo nuove proprietà.
3. Viene ritornato il valore assegnato a **this**.



Costruttore



Quindi

Cosa succede se metto il return nel costruttore?

Come vi ho detto: i costruttori generalmente non hanno l'istruzione **return**. Il loro compito è di eseguire tutto ciò che è necessario a creare l'oggetto lavorando su **this**; quest'ultimo sarà l'output.

Se decidiamo di inserire un'istruzione di **return**, vanno seguite delle semplici regole:

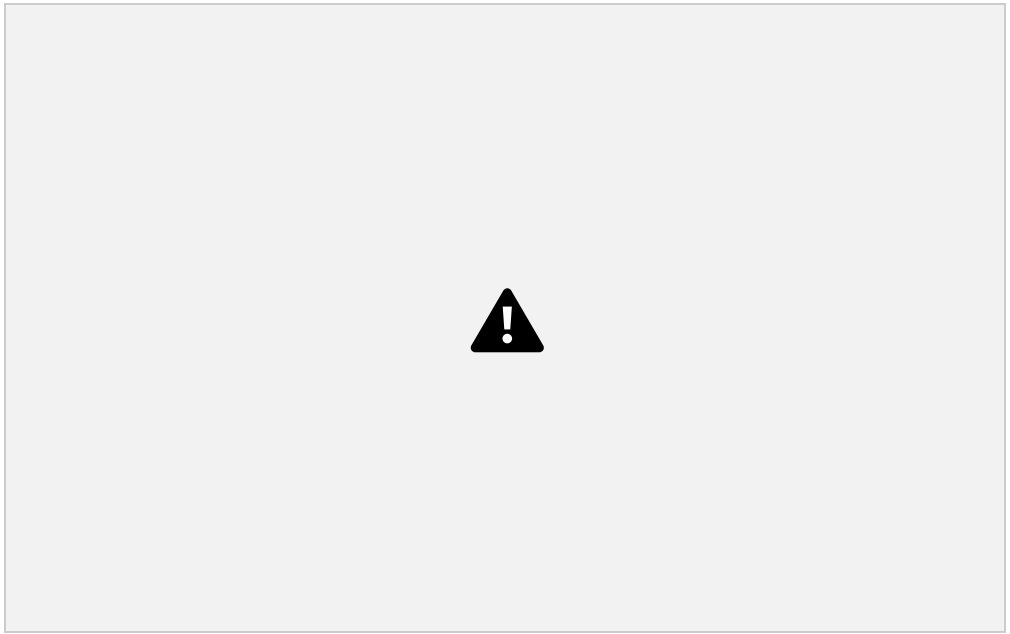
1. Se **return** viene invocato con un oggetto, questo verrà ritornato al posto di this.
2. Se **return** viene invocato con un tipo primitivo, verrà ignorato.

Cosa succede se metto il return nel costruttore?

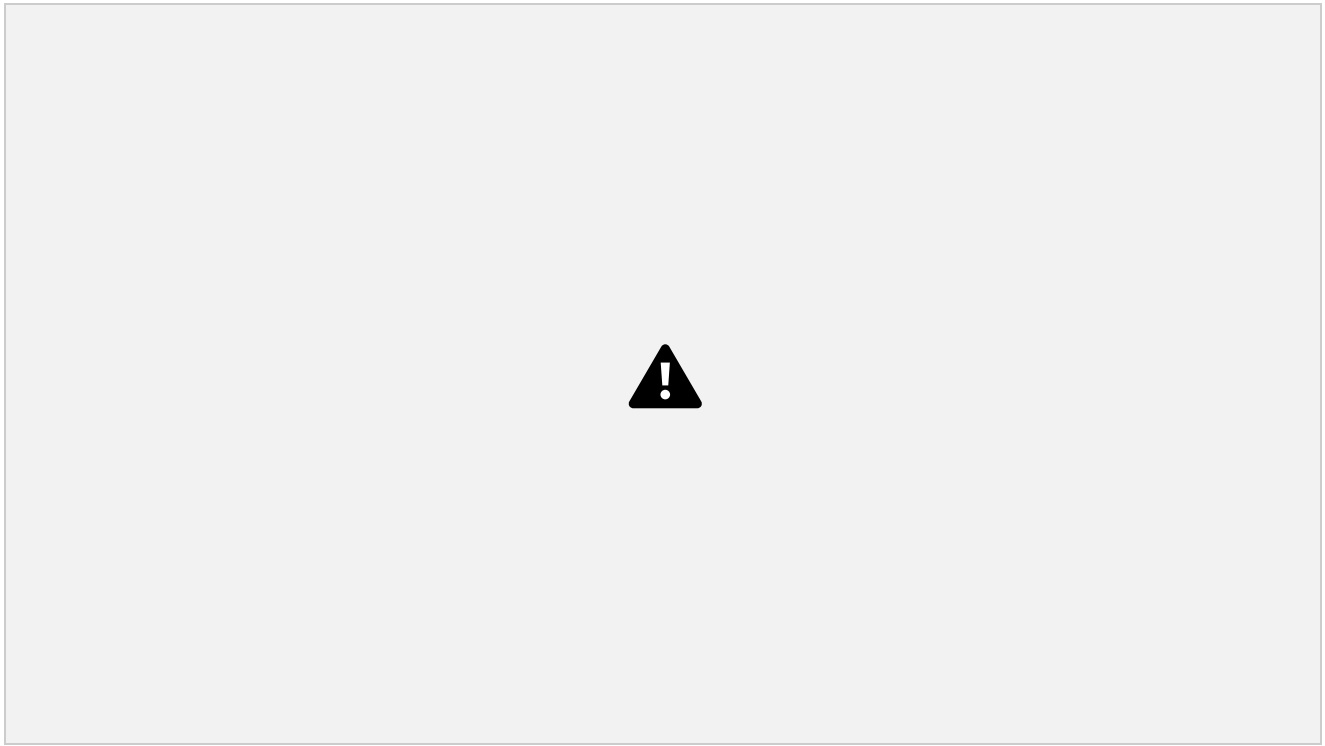
Metodi in un costruttore

Utilizzare costruttori per creare degli oggetti ci dà un grande vantaggio in termini di **flessibilità** e di **riusabilità**. Il costruttore può avere dei parametri che definiscono come costruire l'oggetto, e cosa "*metterci dentro*".

Ovviamente, possiamo aggiunger a **this** non solo **proprietà**, ma anche **metodi**.



Metodi in un costruttore



Domande?

Riepilogo

I costruttori sono solo delle normali funzioni; seguono però una convenzione comune che prevede di denominarle con la prima lettera maiuscola.

Un costruttore dovrebbe essere chiamato solamente utilizzando new. Questo tipo di chiamata implica la creazione di un oggetto vuoto, **this**, che verrà popolato entro la fine della funzione.

Possiamo utilizzare i costruttori **per creare molti oggetti simili tra loro**.

JavaScript fornisce costruttori per la maggior parte degli oggetti integrati nel linguaggio: come **Date** per le date, **Set** per gli insiemi e molti altri.

Esercitazione 1

Scrivi un costruttore Calcolatrice che crea oggetti con 4 metodi:

leggi() richiede due valori come argomento della funzione e li memorizza nelle proprietà dell'oggetto.

somma() ritorna la somma delle proprietà.

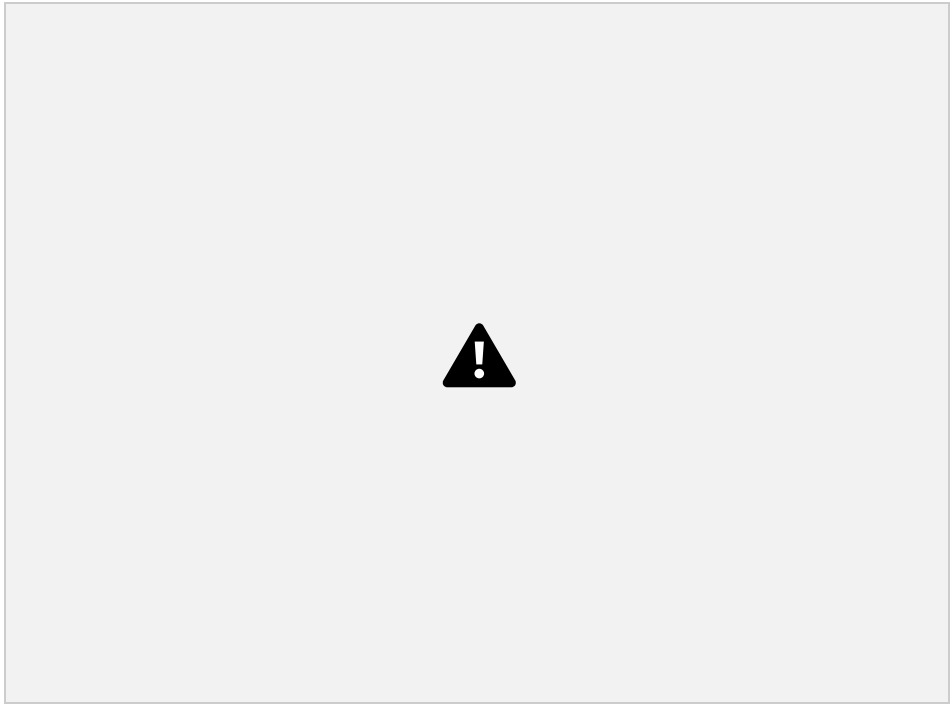
moltiplica() ritorna il prodotto delle proprietà.

dividi() ritorna la divisione delle proprietà

Le Classi

Classe

Nella programmazione orientata agli oggetti una **classe** è un costrutto di un linguaggio di programmazione usato come modello per creare oggetti. Il modello comprende proprietà (attributi) e metodi che saranno condivisi da tutti gli **oggetti creati** (istanze) a partire dalla classe. Un “oggetto” è, di fatto, l’istanza di una classe.

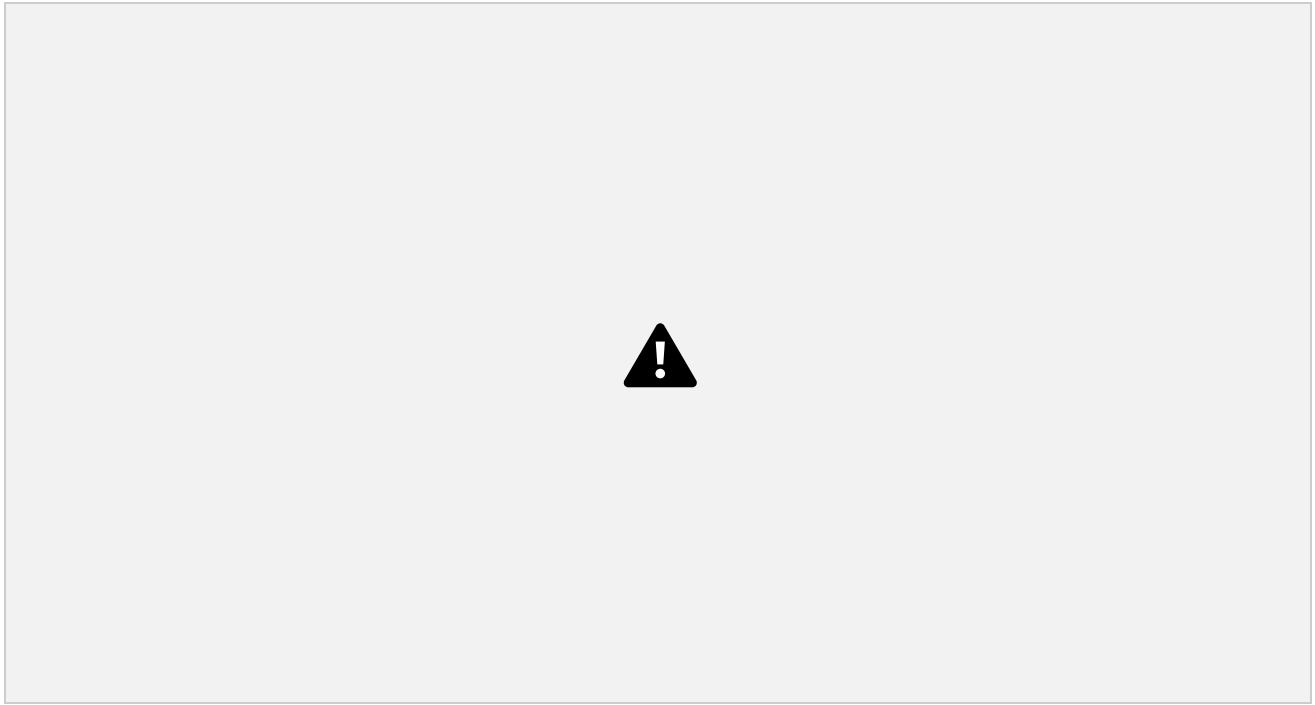


La

sintassi di “class”

Quindi se utilizzassimo **new LaMiaPrimaClasse()** creerà un nuovo oggetto con tutti i **metodi** presenti nella classe.

Il metodo **constructor()** viene chiamato automaticamente da **new**, dunque possiamo usarlo per inizializzare l’oggetto e le sue proprietà.

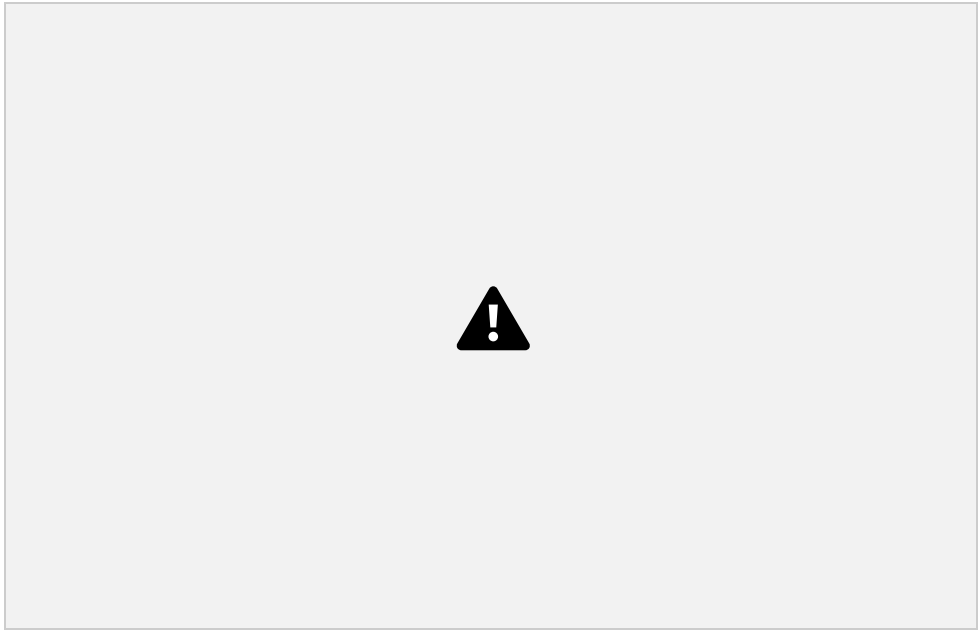


La

sintassi di “class”

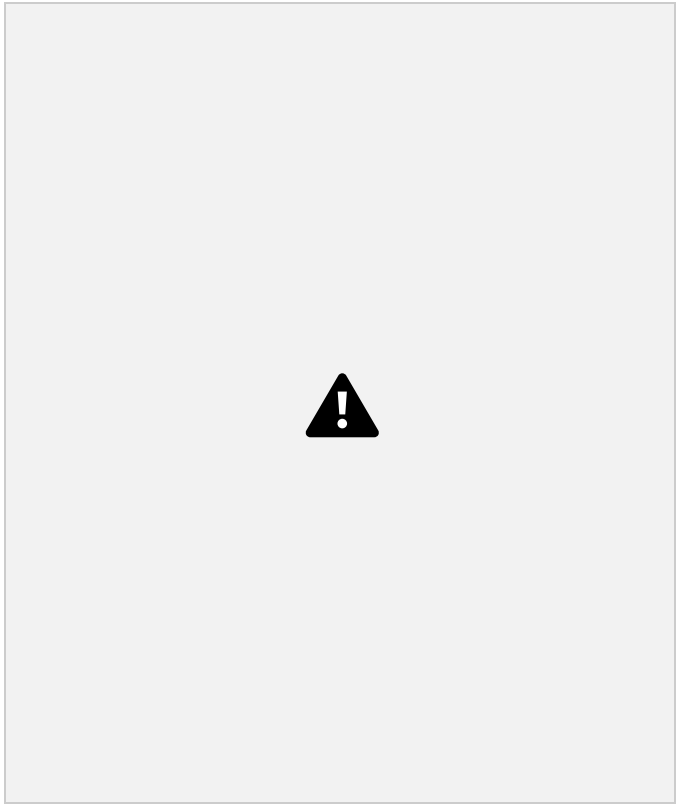
Quando l’interprete di Javascript invoca ***new Cantante("John")***:

- Crea un **nuovo** oggetto;
- Il metodo **constructor()** viene invocato e assegna a **this.nome** l’argomento dato in input al costruttore.
- Successivamente si possono invocare i metodi della classe, ad esempio **cantante.cantaUnaCanzone()**.

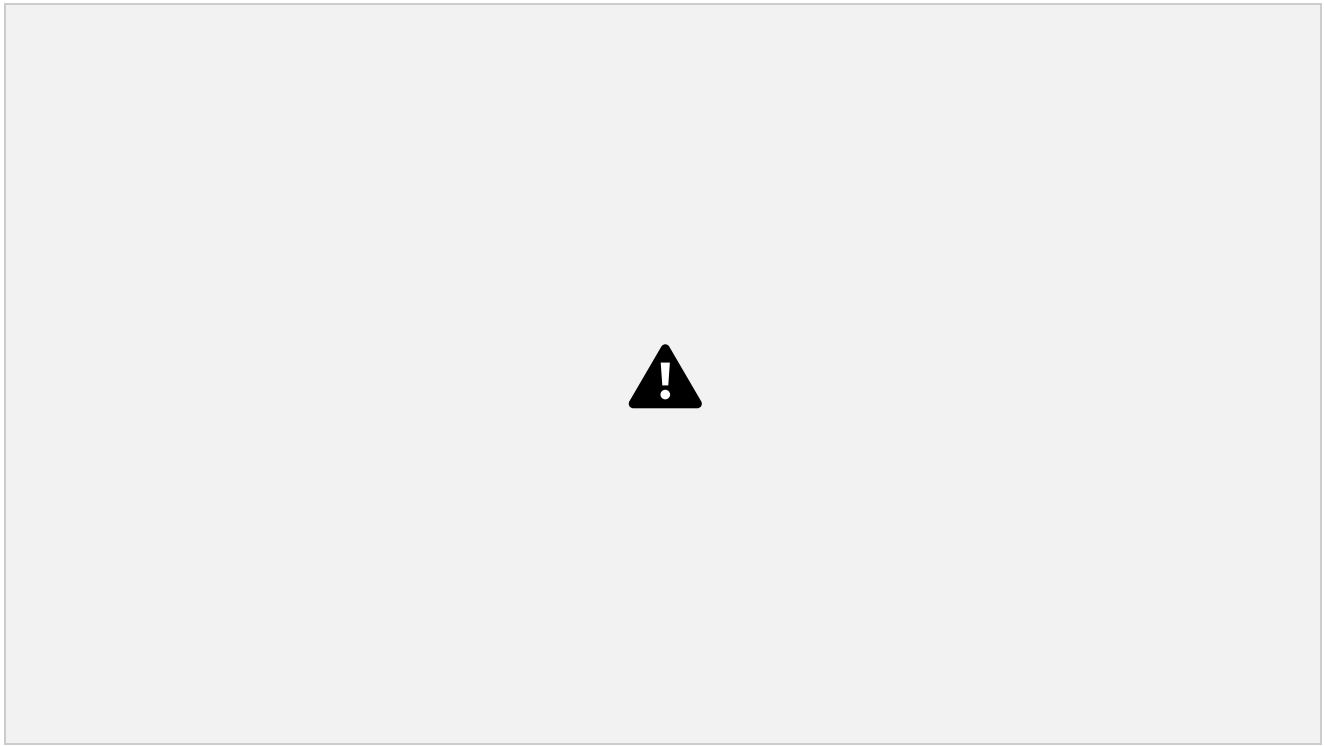


Proprietà di una classe

Possiamo creare una proprietà di una classe anche utilizzando l’operatore **=**.



Getters e Setters



Domande?

Esercitazione 1

Scrivere la classe Calcolatrice che contiene al suo interno due proprietà di tipo intero parametro1 e parametro2 che non possono essere inferiore a zero. La classe dovrà avere anche tre metodi:

somma() ritorna la somma delle proprietà.

moltiplica() ritorna il prodotto delle proprietà.

dividi() ritorna la divisione delle proprietà

Prossima lezione

- Ereditarietà e metodi statici
- Le Clousure e scope
- Le funzioni lambda o arrows function