Javascript

Corso Sviluppatore front end

Docente: Ignazio Sammarco

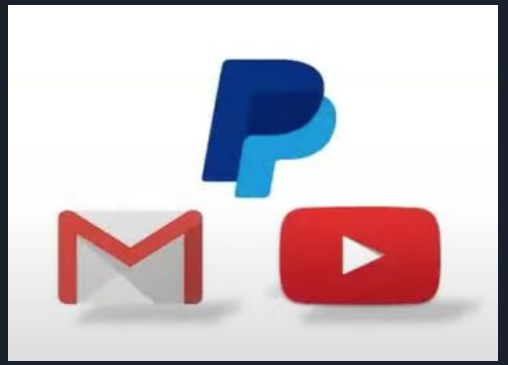
Arces Formazione 2022

Javascript

- ★ Sviluppato nel 1995 in soli 10 giorni da Brendan Eich
- ★ Inizialmente chiamato Mocha, poi LiveScript e infine Javascript
- ★ Creato per aggiungere funzionalità dinamiche alle pagine HTML
- ★ Nessun legame con Java (sintassi diverse)

Javascript

Website & Application



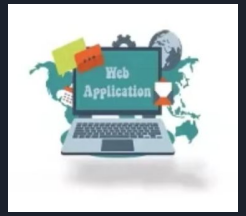
Industry

Javascript


Scripting Language
Client-side Technology
User Validation
Object oriented
JS
Light-weight
Server-side Technology
Platform independent
Case Sensitive

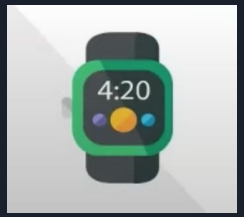
Applicazioni di Javascript

Website Development



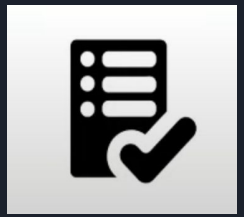
Web Application

Game



Smartwatch

Mobile application



Input validation

Applicazioni di Javascript

Javascript è un linguaggio interpretato:

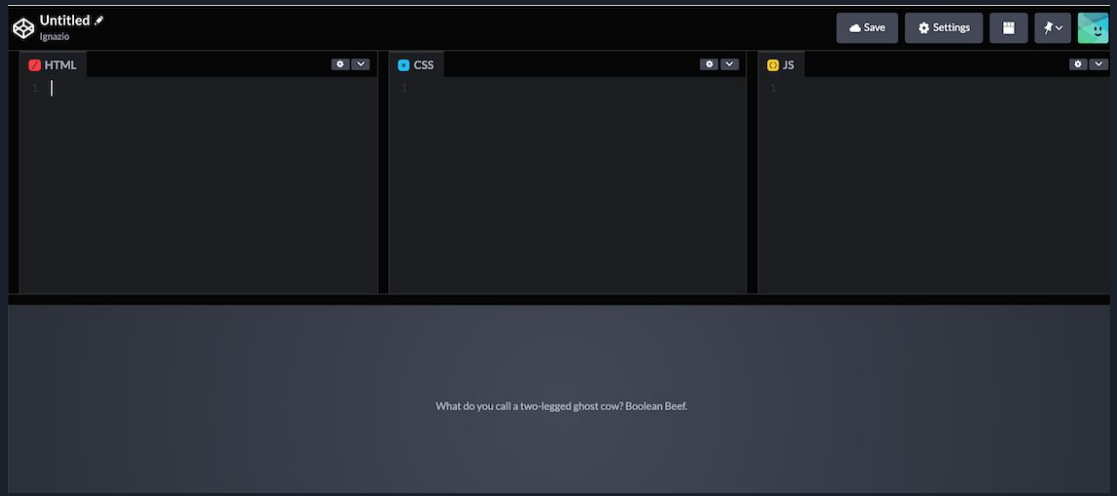
Con questo vogliamo dire che il codice che scriviamo non viene preventivamente compilato (come si fa con Java) e poi eseguito.

Un file javascript si scrive come file sorgente e quindi è eseguito direttamente dal browser.

Il motore javascript (engine) del browser provvederà a interpretarlo in tempo reale e a trasformarlo in linguaggio comprensibile dalla macchina (linguaggio macchina).

Applicazioni di Javascript

Ma facciamo un esempio di codice javascript.
Per far questo ci affidiamo ad un sito Codepen.io dove scriveremo il nostro primo codice sorgente.



Editor online

https://app.codingrooms.com/

https://app.codingrooms.com/w/pywqyWr5PaYu

https://jsfiddle.net/

https://codesandbox.io/?from-app=1

https://jseditor.io/

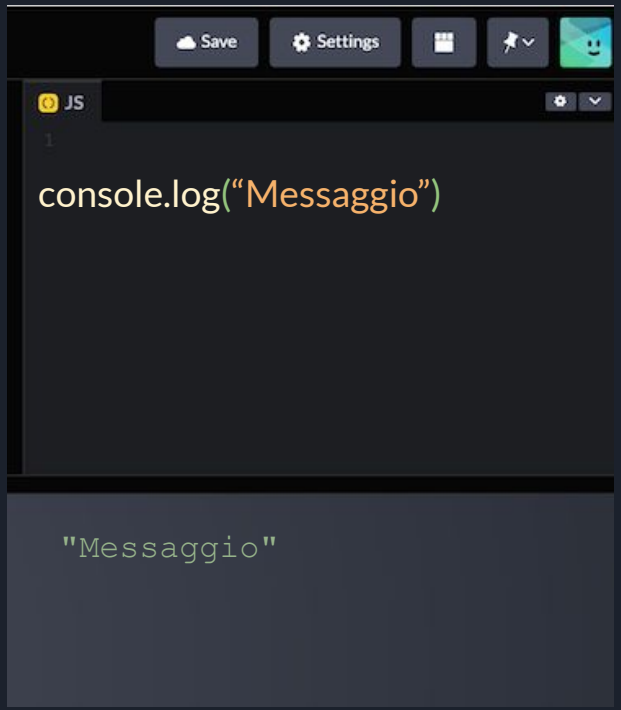
https://stackblitz.com/

Applicazioni di Javascript

Usiamo la funzione

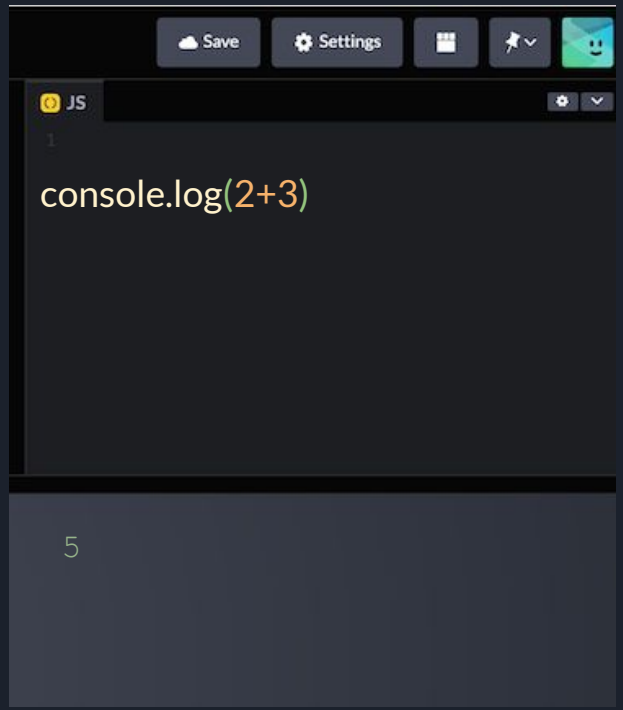
console.log()

Con questa funzione si scrive un messaggio sulla console del browser.



Applicazioni di Javascript

Ma si possono anche inserire espressioni numeriche come somme, moltiplicazioni ecc.



Applicazioni di Javascript

Si possono anche combinare testo ed espressioni numeriche nello stesso messaggio.

```
console.log("la somma è: " + 2 + 3);
```



```
a = 2 + 3;
console.log("la somma è: " + a);
```

Interazione in Javascript

Javascript

Javascript ci permette di interagire con un utente mentre naviga una pagina con alcune funzioni di base

alert()

prompt()

confirm()

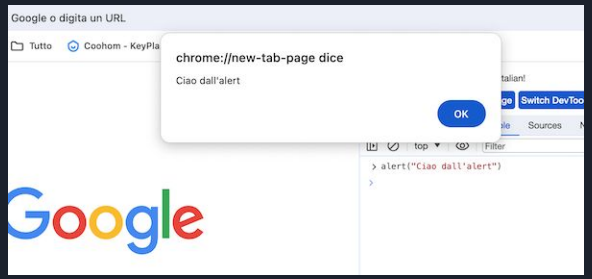
Javascript

alert()

Con questa funzione viene mostrato una finestra con un messaggio e si resta in attesa che l'utente pressi il tasto ok.

Esempio

```
alert("Ciao dall'alert")
```



Javascript

prompt()

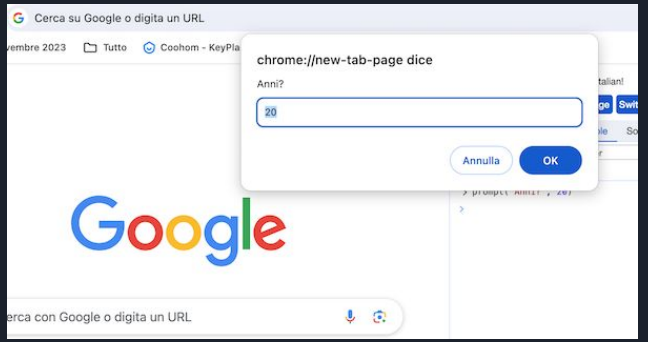
La funzione prompt accetta due argomenti

prompt(title, [default]);

mostra una finestra modale con un campo di input e due pulsanti

Esempio

```
result = prompt("Anni?", 20 )
```

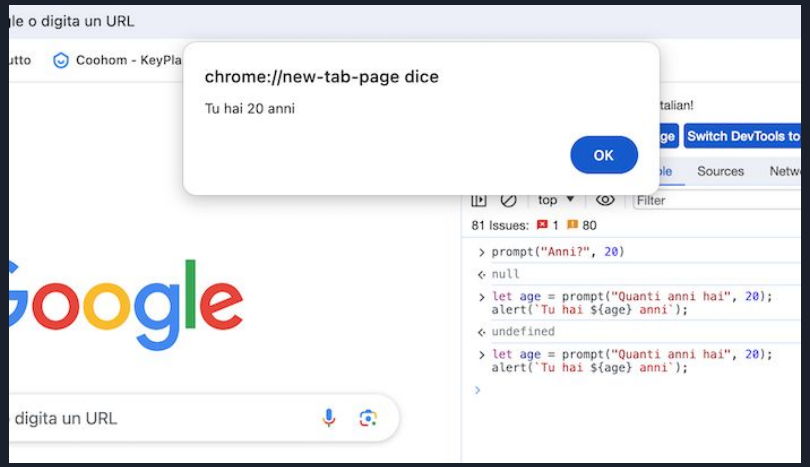


Javascript

prompt()

Il valore immesso nella casella di input del prompt può essere quindi usato da altre funzioni.

```
age = prompt("Quanti anni hai", 20);
alert(`Tu hai ${age} anni`)
```



Javascript

confirm()

Mostra una finestra modale con una domanda e due pulsanti: OK e Cancel

result = confirm(question);

Se premuto ok result è True altrimenti è False.

Esempio

```
confirm("Sei italiano?")
```

Variabili in Javascript

Variabili in Javascript

Possiamo definire una variabile in vari modi

Con la parola chiave var

```
var nominativo;
```

Senza la parola chiave var MA assegnando subito un valore

```
nominativo = "Franco";
```

Variabili in Javascript

Possiamo definire una variabile anche con let

```
let nominativo = "Franco"
```

```
if (true) {
  let y = 30;
}

console.log(y); // y è undefined
```

La visibilità della variabile è limitata all'interna del blocco if{ }

Variabili in Javascript

Si può anche assegnare lo stesso valore a più variabili contemporaneamente:

```
nome = cognome = "Andrea"
```

Variabili in Javascript

Una variabile che è stata dichiarata ma a cui non è assegnato nessun valore è undefined

```
var nome;
tipo = typeof(nome)
console.log(tipo)
```

Variabili in Javascript

Le variabili definite al di fuori di una funzione sono globali.

Quindi sono viste **dappertutto**, anche all'interno della funzione.

In questo caso la funzione incrementa() saprà quanto vale la variabile x e aggiungerà a 10 il numero 5.

```
var eta = 10;
incrementa(5);
function incrementa(valore){
    eta = eta + valore;
}
console.log(eta) // eta sarà 15
```

Variabili in Javascript

Ma se definiamo una variabile locale (dentro la funzione) con lo stesso nome di quella globale (in questo caso x), la variabile presente all'interno della funzione avrà priorità su quella all'esterno.

```
var x = 10;
incrementa(5);
function incrementa(valore) {
     var x = 1;
     x = x + valore;
}
console.log(x);      // x avrà valore 6
```

Variabili in Javascript

Dichiarare una variabile all'interno di un blocco di codice non crea un nuovo *scope* per la variabile! La variabile sarà visibile anche fuori tranne se non si usa let.

```
var x = 10;
var y;
{
  let x = 20;
  y = x + 1;
}
console.log(x); //x sarà 10
console.log(y); //x sarà 21
```

Tipi di dati in Javascript

I tipi di dati in Javascript possono essere:

- string
- number
- boolean
- object
- undefined
- null

Tipi di dati in Javascript

In questo caso la nostra variabile `nome` sarà di tipo string.

Questo perché il tipo sarà inferito al momento dell'assegnazione del valore.

```
nome = "Andrea"
```

Tipi di dati in Javascript

Per conoscere il tipo di una variabile possiamo usare la funzione javascript typeof()

```
tipo = typeof(nome)
console.log(tipo)
```

Console

```
> typeof(a)
< 'string'
```

Tipi di dati in Javascript

Possiamo anche concatenare due o più stringhe.

Concateniamo la variabile nome con la variabile cognome e una stringa fissa.

```
nome = "Andrea"
cognome = "Belli"
nominat = "Mi chiamo "+nome+" "+cognome


console.log(nominat)
```

Tipi di dati in Javascript

L'operatore `+=` si usa per aggiungere una stringa ad un'altra stringa in modo abbreviato.

```
nominativo = nominativo + " da Roma !"
nominativo += " da Roma !"

console.log(nominativo)
```

Tipi di dati in Javascript

Ma si può usare anche per i numeri.

```
eta = 20
eta += 4

console.log(eta)
```

Tipi di dati in Javascript

Per convertire da un tipo ad un altro (ad es numero->stringa) ci sono diversi modi:

Si possono usare le funzioni String() e toString().

Esiste anche la conversione implicita

```
n = 234;
var s = String(n);
var s = n.toString();

// ma anche conversione implicita
var s = "" + 234
console.log(typeof(s))
```

Tipi di dati in Javascript

Mettendo un segno “+” davanti ad una stringa contenente solo numeri si converte la stringa in number.

```
//conversione da stringa a numero

var s = +"234"
console.log(typeof(s)) // number
```

Condizioni

In Javascript, come in altri linguaggi, esistono gli operatori condizionali.

Il più famoso è l'operatore if.

```
if (age>17) {
 console.log("Maggiorenne");
}else{
 console.log("Minorenne");
}
```

Condizioni

Un operatore simile all'if è l'operatore ternario .

```
age = 20
console.log( age > 17 ? "Maggiorenne" : "Minorenne" )
```

Se l'età è maggiore di 17 la console stamperà la stringa "Maggiorenne " altrimenti stamperà la stringa "Minorenne"

Tipi di dati in Javascript

https://codepen.io/uniqname/pen/elApt

```
function square(numero){

}
```

Tipi di dati in Javascript

Array

Array in Javascript

Un array è una variabile che contiene più valori raggruppati sotto un unico nome.

In questo caso l'array è animali e al suo interno contiene tre valori (Cane, Gatto e Cavallo).

Codice

```
const animali = ["Cane", "Gatto", "Cavallo"];

console.log(animali);
```

Output

```
(3) ['Cane', 'Gatto', 'Cavallo']
    0: "Cane"
    1: "Gatto"
    2: "Cavallo"
    length: 3
    [[Prototype]]: Array(0)
```

Array in Javascript

Per accedere ai valori contenuti nell'array basta indicare il nome dell'array seguito dall'indice o posizione in cui si trova quel valore.

Il valore "Cane" in questo caso si trova in posizione o indice 0 dell'array animali.

Codice

```
const animali = ["Cane", "Gatto", "Cavallo"];

console.log(animali[0]);
console.log("Specie: "+animali[1]);
```

Output

```
"Cane"
"Specie: Gatto"
```

Array in Javascript

Array in Javascript

Creare un array

1° MODO

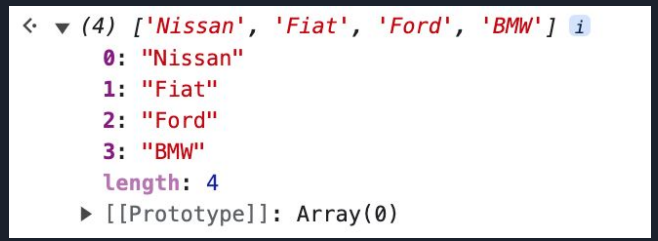
E’ possibile inserire i valori dell’array dentro le parentesi quadre [] separati dalla virgola “,”

Codice

```
let cars = ["Nissan", "Fiat", “Ford”, “BMW”];

console.log(cars);
```

Output



Array in Javascript

Creare un array

2° MODO

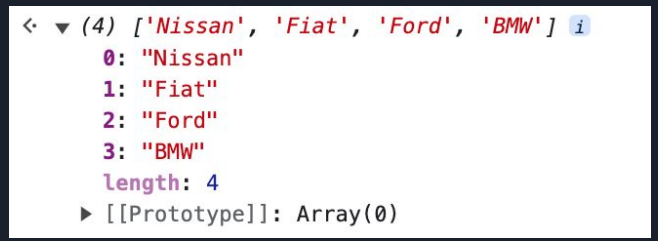
E’ possibile creare un array attraverso un costruttore che istanzia un oggetto di tipo Array.

Codice

```
let cars = new Array("Nissan", "Fiat", "Ford",
"BMW");

console.log(cars);
```

Output



Array in Javascript

Inserire elementi

Una volta creato l'array si possono inserire nuovi elementi con le funzioni:

- push()

Inserisce l'elemento alla FINE

Codice

```
cars.push()("Mercedes");

console.log(cars);
```

Output

```
(3) ['Nissan', 'Fiat', 'Ford', 'BMW', 'Mercedes']
    0: "Nissan"
    1: "Fiat"
    2: "Ford"
    3: "BMW"
    4: "Mercedes"
    length: 5
    [[Prototype]]: Array(0)
```

Array in Javascript

Inserire elementi

Una volta creato l'array si possono inserire nuovi elementi con le funzioni:

- unshift()

Lo inserisce all' INIZIO

Codice

```
cars.unshift()("FERRARI");

console.log(cars);
```

Output

```
(3) ['FERRARI', 'Nissan', 'Fiat', 'Ford', 'BMW', 'Mercedes']
    0: "FERRARI"
    1: "Nissan"
    2: "Fiat"
    3: "Ford"
    4: "BMW"
    5: "Mercedes"
    length: 6
    [[Prototype]]: Array(0)
```

Array in Javascript

Rimuovere elementi

Da un array si possono rimuovere deglii elementi con le funzioni:

- pop()

Rimuove l'elemento finale

Codice

```
cars.pop();

console.log(cars);
```

Output

```
(3) ['FERRARI', 'Nissan', 'Fiat', 'Ford', 'BMW']
    0: "FERRARI"
    1: "Nissan"
    2: "Fiat"
    3: "Ford"
    4: "BMW"
    length: 5
    [[Prototype]]: Array(0)
```

Array in Javascript

Rimuovere elementi

Da un array si possono rimuovere degli elementi con le funzioni:

- shift()

Rimuove il primo elemento

Codice

```
cars.shift()();

console.log(cars);
```

Output

```
(3) ['Nissan', 'Fiat', 'Ford', 'BMW']
    0: "Nissan"
    1: "Fiat"
    2: "Ford"
    3: "BMW"
    4: "Mercedes"
    length: 5
    [[Prototype]]: Array(0)
```

Array in Javascript

Ma se abbiamo molti valori possiamo accedere anche attraverso un ciclo for.

Il ciclo valuterà tutte le posizioni dell'array ed eseguira per ogni posizione il codice inserito all'interno del blocco di codice for.

Codice

```
const animali = ["Cane", "Gatto"];

for (let index=0; i<animali.length; index++ ) {
  console.log("Specie: "+animali[index]);
}
```

Output

```
"Specie: Cane"
"Specie: Gatto"
```

Array in Javascript

Il ciclo for può essere scritto in altra forma.

Per tutti gli indici presenti nell'array sarà stampato in console il valore.

Codice

```
const animali = ["Cane", "Gatto"];

for (index in animali) {
  console.log("Specie: "+animali[index]);
}
```

Output

```
"Specie: Cane"
"Specie: Gatto"
```

Array in Javascript

E’ possibile anche eseguire la stessa operazione su tutti gli elementi di un array attraverso la funzione map().

Nell’esempio la funzione x=> x*2, passata a map(), raddoppierà ogni elemento dell’array.

Codice

```
const arr_1 = [1, 4, 9, 16];

const map1 = arr_1.map(x => x * 2);

console.log(map1);
```

Output

```
Array [2, 8, 18, 32]
```

Array in Javascript

ESERCIZIO

Dato un array di numeri noti

const array = [2, 5, 3, 8]

Creare un altro array costituito dai quadrati degli stessi numeri

const array_2 = [4, 25, 9, 64]

SOLUZIONE

```
const array = [2, 5, 3, 8];
const array_2 = array.map(
     x => x * x
);
console.log(array_2);
```

Array in Javascript

ESERCIZIO

Dato un array di nomi di calciatori

const array = [“totti, “ronaldo”, “messi”]

Creare un altro array dove la prima lettera di ogni giocatore è maiuscola

const array 2= [“Totti”, “Ronaldo”, “Messi”]

SOLUZIONE

```
const array = [”totti”, ”ronaldo”,”messi”];

const array2 = array.map(

   nome => nome[0].toUpperCase()+nome.slice(1)

);

console.log(array2);
```

Oggetti in Javascript

Un oggetto in javascript può essere definito in diversi modi:

- LITERAL
- CONSTRUCTOR
- PROTOTYPES

Oggetti in Javascript

Un oggetto LITERAL può essere definito come un insieme di coppie CHIAVE - VALORE racchiuse all'interno di due parentesi graffe e separate dalle virgole.

Codice

```
const persona = {
      nome: "Marco",
      età: 25
}
console.log(persona);
```

Output

```
{nome: "Marco", età: 25}
1.    età: 25
2.    nome: "Marco"
3.    __proto__: Object
```

Oggetti in Javascript

Per accedere ad un determinato valore bisogna indicare il nome dell'oggetto seguito da un punto e la chiave.

Codice

```
var nome = persona.nome;
var età = persona.età;

console.log("Nome: " + nome);
console.log("Età: " + età);
```

Output

```
"Nome: Marco"
"Età: 25"
```

Oggetti in Javascript

Gli oggetti literal in JavaScript sono entità dinamiche, la loro struttura può essere modificata anche dopo averla definita.

Codice

```
var persona = {};
persona.nome = "Mario";
persona.cognome = "Rossi";
persona.indirizzo = {
 via: "Via Garibaldi",
 numero: 15,
 CAP: "00100",
 citta: "Roma"
};
persona.eta = 32;
console.log(persona)
```

Output

```
// [object Object]
{
  "nome": "Mario",
  "cognome": "Rossi",
  "indirizzo": {
    "via": "Via Garibaldi",
    "numero": 15,
    "CAP": "00100",
    "citta": "Roma"
  },
  "eta": 32
}
```

Oggetti in Javascript

Un oggetto può contenere anche delle funzioni (metodi) al suo interno, oltre le proprietà.

In questo caso abbiamo un metodo calcolaEta()

Codice

```
const persona = {
 nome: 'Marco',
 anno_nasc: 1998,
 calcolaEta: function () {
   const eta = new Date().getFullYear() -
this.anno_nasc;
   return `L'età di ${this.nome} è ${eta} anni`;
 },
};
console.log(persona.calcolaEta());
```

Oggetti in Javascript

Ecco un altro esempio di metodo che restituisce il nominativo completo di Nome e Cognome, a partire dalle informazioni separate.

Codice

```
persona = {
 nome: "Mario",
 cognome: "Rossi",
 getNominativo: function() {
     return this.nome+this.cognome;
 }
}
console.log(persona.getNominativo());
```

Oggetti in Javascript

La funzione Object.values() restituisce un array dei valori delle proprietà dell'oggetto passato.

Codice

```
console.log(Object.values(persona));
console.log(valori)
```

Console browser

```
["Mario", "Rossi", ƒ]
    0: "Mario"
    1: "Rossi"
    2: ƒ getNominativo()
```

Oggetti in Javascript

Se volessimo creare più oggetti con la notazione Literal (parentesi graffe) dobbiamo ripetere il metodo getNominativo() più volte.

```
persona = {
 nome: "Mario",
 cognome: "Rossi",
 getNominativo: function() {
     return this.nome+this.cognome;
 }
}

persona = {
 nome: "Marco",
 cognome: "Bianchi",
 getNominativo: function() {
     return this.nome+this.cognome;
 }
}
```