



FONDAMENTI DI INFORMATICA

Alma Artis

Francesca Pratesi (ISTI, CNR)

Visibilità delle variabili e stato

ESERCIZIO - 1

- Si scriva un funzione in JavaScript con un parametro numerico che rappresenta la velocità di un veicolo in km orari. La funzione deve calcolare e restituire in output l'eventuale multa (in euro) da infliggere al conducente, secondo la normativa vigente sulle autostrade che risulta dalla seguente tabella:

Velocità v	Multa
$v > 200$	1000
$160 < v \leq 200$	500
$130 < v \leq 160$	200
$v \leq 130$	0

- Si completi il programma invocando la funzione precedentemente definita per ottenere la multa da infliggere ad un conducente che viaggia ad una velocità ricevuta in input dall'utente, stampando quindi un messaggio che comunichi la sanzione inflitta al conducente.

ESERCIZIO - 2

- Dato l'esercizio della lezione scorsa (quello del grado alcolico) risolvere il problema con uno switch.
- Nota: non occorre inserire tutti i valori, ma attenersi a questa nuova tabella

Grado alcolico (g)	Messaggio
$g > 40 \ \&\& \ g < 45$	Superalcolico
$g == 20 \ \ g == 40$	Alcolico
$15 < g \leq 20$	Vino liquoroso
$13 < g \leq 15$	Vino forte
$10 < g \leq 13$	Vino normale
$g \leq 10$	Poco alcolico

ISTRUZIONE SWITCH CON INTERVALLI

```
var temperatura = -43;
switch (true) {
  case temperatura>30:
    console.log('Molto caldo');
    break;
  case temperatura>20:
    console.log('Caldo');
    break;
  case temperatura>10:
    console.log('Gradevole');
    break;
  case temperatura>0:
    console.log('Freddo');
    break;
  default:
    console.log('Molto freddo');
}
```



ESERCIZIO - 3

- Scrivere un programma che chieda in input tre valori che rappresentano le lunghezze di tre lati di un triangolo; decidere se il triangolo è equilatero, isoscele o scaleno e stampare il risultato sulla console



ESERCIZIO - 3

- Scrivere un programma che chieda in input tre valori che rappresentano le lunghezze di tre lati di un triangolo; decidere se il triangolo è equilatero, isoscele o scaleno e stampare il risultato sulla console
 - *Idea 1: confrontare i lati a coppie fino a quando non si hanno informazioni sufficienti a decidere di che tipo è il triangolo*
 - *Idea 2: contare quante coppie di lati uguali ci sono e in base a questo identificare il tipo di triangolo*



ESERCIZI - 4

- Operatore XOR
 - L'operatore booleano binario $\text{XOR}(a,b)$ calcola:
 - **true**, quando solamente uno tra a e b è true,
 - **false**, in tutti gli altri casi.
 - Si scriva un programma che definisca una funzione XOR che, avendo come parametri due valori booleani a e b , restituisca il risultato di $\text{XOR}(a,b)$.
Si completi il programma invocando la funzione precedentemente definita per tutte le possibili combinazioni di valori booleani per i suoi parametri, e si stampino i risultati ottenuti.
- Si definisca una funzione XOR_senza_if, con funzionamento simile alla funzione XOR (stesso input e output) ma che non faccia uso dell'istruzione if.



LIBRI E RIFERIMENTI

- Capitolo 2

Eloquent Javascript – Second Edition

Marijn Haverbeke

Licensed under CC license.

Available here: <http://eloquentjavascript.net/>



STRUTTURA DI UN PROGRAMMA JAVASCRIPT

IMPORTANTE

- I nomi dei parametri formali e dei parametri attuali **possono** essere diversi

```
function calcola_somma(n,m){  
    return (n+m);  
}  
// corpo del programma  
var addendo1=2, addendo2=3, n=5, m=7;  
console.log(calcola_somma(addendo1,addendo2));  
console.log(calcola_somma(n,m));
```



AMBIENTE E VISIBILITÀ DELLE VARIABILI

- Un ambiente è costituito da un insieme di variabili e di funzioni
 - in un ambiente non possono esserci due variabili o due funzioni con lo stesso nome (identificatore)
- Un ambiente è modificato:
 - dalle dichiarazioni: aggiungono una variabile o una funzione all'ambiente
 - dai comandi che modificano i valori delle variabili
- Ogni variabile è visibile nei punti del programma in cui fa parte dell'ambiente
 - Anche i blocchi racchiusi tra { } definiscono ambienti diversi



VARIABILI GLOBALI E LOCALI

- Una variabile globale è visibile in tutto il programma
 - A meno che non vengano mascherate
- Una variabile locale è visibile solo in alcuni punti del programma
- Per i nostri scopi un parametro formale si comporta come una variabile locale
 - NB: il parametro formale non deve essere dichiarato dalla keyword var



VARIABILI GLOBALI E LOCALI - ESEMPIO

```
function funzione_prova(numero){  
    var nome = 'Francesca';  
    var testo = '';  
    var b = 5;  
    if (numero>0) testo = nome;  
    else testo = ''+b;  
    return testo;  
}
```

```
var a, b, x;  
a = 12;  
b = 45;  
x = 2**3;  
console.log(funzione_prova(a));
```




VARIABILI GLOBALI E LOCALI - ESEMPIO

```
function funzione_prova(numero){  
    var nome = 'Francesca';  
    var testo = '';  
    var b = 5;  
    if (numero>0) testo = nome;  
    else testo = ''+b;  
    return testo;  
}
```

variabili globali

```
var a, b, x;  
a = 12;  
b = 45;  
x = 2**3;  
console.log(funzione_prova(a));
```



VARIABILI GLOBALI E LOCALI - ESEMPIO

```
function funzione_prova(numero){  
  var nome ← 'Francesca';  
  var testo ← '';  
  var b = 5;  
  if (numero>0) testo = nome;  
  else testo = ''+b;  
  return testo;  
}  
  
var a, b, x;  
a = 12;  
b = 45;  
x = 2**3;  
console.log(funzione_prova(a));
```

variabili locali

variabili globali

SHADOWING O NAME MASKING

- Una variabile globale può essere nascosta da una variabile locale che ha lo stesso identificatore
 - una variabile nascosta non è accessibile



NAME MASKING - ESEMPIO

```
function funzione1(){  
  var x, y;  
  x = 12;  
  y = 25;  
}  
function funzione2(){  
  var x, a;  
  x = 32;  
  a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;
```



NAME MASKING - ESEMPIO

```
function funzione1(){  
  var x, y;  
  x = 12;  
  y = 25;  
}  
function funzione2(){  
  var x, a;  
  x = 32;  
  a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;
```

In funzione1 sono accessibili le variabili:

- locali: x e y
- globali: a e b

Le variabili globali x e y non sono visibili!



NAME MASKING - ESEMPIO

```
function funzione1(){  
  var x, y;  
  x = 12;  
  y = 25;  
}  
function funzione2(){  
  var x, a;  
  x = 32;  
  a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;
```

In funzione2 sono accessibili le variabili:

- locali: x e a
- globali: b e y

Le variabili globali a e x non sono visibili!



NAME MASKING - ESEMPIO

```
function funzione1(){  
  var x, y;  
  x = 12;  
  y = 25;  
}  
function funzione2(){  
  var x, a;  
  x = 32;  
  a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;
```

Nel corpo del programma sono accessibili le variabili:

- globali: a, b, x e y

Le variabili locali a funzione1 e funzione2 non sono visibili!



LO STATO E LE VARIABILI DI AMBIENTE

- Come già detto, lo stato viene generato a partire dalle funzioni e dalle variabili accessibili in un particolare punto del programma



LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;  
    x = 12;  
    y = 25;  
}  
function funzione2(){  
    var x, a;  
    x = 32;  
    a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
funzione1();
```



LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){
    var x, y;
    x = 12;
    y = 25;
}
function funzione2(x){
    var a;
    x = 32;
    a = 42;
}
// corpo del programma
var a, b, x, y;
a = 12;
b = 45;
x = 42;
y = 2**3;
funzione1();
```

← {{funzione1, function(){...}}}

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;  
    x = 12;  
    y = 25;  
}  
function funzione2(x){  
    var a;  
    x = 32;  
    a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
funzione1();
```

← {(funzione1, function(){...}), (funzione2, function(x){...})}

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){
    var x, y;
    x = 12;
    y = 25;
}
function funzione2(x){
    var a;
    x = 32;
    a = 42;
}
// corpo del programma
var a, b, x, y;
a = 12;
b = 45;
x = 42;
y = 2**3;
funzione1();
```


{(a,undefined), (b,undefined), (x,undefined), (y,undefined), (funzione1, function(){...}), (funzione2, function(x){...})}

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;  
    x = 12;  
    y = 25;  
}  
function funzione2(x){  
    var a;  
    x = 32;  
    a = 42;  
}
```

// corpo del programma

```
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
funzione1();
```

 `{{(a,12), (b,undefined), (x,undefined), (y,undefined), (funzione1, function(){...}), (funzione2, function(x){...})}}`

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;  
    x = 12;  
    y = 25;  
}  
function funzione2(x){  
    var a;  
    x = 32;  
    a = 42;  
}
```

// corpo del programma

```
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
funzione1();
```

 {(a,12), (b,45), (x,undefined), (y,undefined), (funzione1, function(){...}), (funzione2, function(x){...})}

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){
    var x, y;
    x = 12;
    y = 25;
}
function funzione2(x){
    var a;
    x = 32;
    a = 42;
}
// corpo del programma
var a, b, x, y;
a = 12;
b = 45;
x = 42;
y = 2**3;
function1();
```

{(a,12), (b,45), (x,42), (y,undefined), (funzione1, function(){...}), (funzione2, function(x){...})}

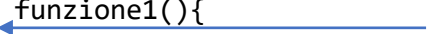
LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;  
    x = 12;  
    y = 25;  
}  
function funzione2(x){  
    var a;  
    x = 32;  
    a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
function1();
```

$\{(a,12), (b,45), (x,42), (y,8), (funzione1, function()\{...\}), (funzione2, function(x)\{...\})\}$

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){
  var x, y;
  x = 12;
  y = 25;
}
function funzione2(x){
  var a;
  x = 32;
  a = 42;
}
// corpo del programma
var a, b, x, y;
a = 12;
b = 45;
x = 42;
y = 2**3;
funzione1();
```



LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;                                     {(x,undefined), (y,undefined), (a,12), (b,45), (funzione1, function(x){...}), (funzione2,  
    x = 12;                                       function(){...})}  
    y = 25;  
}  
function funzione2(x){  
    var a;  
    x = 32;  
    a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
funzione1();
```

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;  
    x = 12; ————— {(x,12), (y,undefined), (a,12), (b,45), (funzione1, function(){...}), (funzione2, function(x){...})}  
    y = 25;  
}  
function funzione2(x){  
    var a;  
    x = 32;  
    a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
funzione1();
```


LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){
    var x, y;
    x = 12;
    y = 25;
}
function funzione2(x){
    var a;
    x = 32;
    a = 42;
}
// corpo del programma
var a, b, x, y;
a = 12;
b = 45;
x = 42;
y = 2**3;
function1();
```

Diagram illustrating the environment state (environment frame) for the function `funzione1` at the time of its execution. The frame contains the following entries:

- `(x,12)`
- `(y,25)`
- `(a,12)`
- `(b,45)`
- `(funzione1, function(){...})`
- `(funzione2, function(x){...})`

A blue arrow points from the `y = 25;` line in the `funzione1` function definition to the `(y,25)` entry in the environment frame.

LO STATO E LE VARIABILI DI AMBIENTE - ESEMPIO

```
function funzione1(){  
    var x, y;  
    x = 12;  
    y = 25;  
}  
function funzione2(x){  
    var a;  
    x = 32;  
    a = 42;  
}  
// corpo del programma  
var a, b, x, y;  
a = 12;  
b = 45;  
x = 42;  
y = 2**3;  
function1();
```

← {(a,12), (b,45), (x,42), (y,8), (funzione1, function(){...}), (funzione2, function(x){...})}