Tecnologie Cloud e Mobile

Lez. 02

JavaScript, Elementi Base

Giuseppe Psaila

Università di Bergamo

giuseppe.psaila@unibg.it

JavScript

- Fin dai primi anni di vita del World Wide Web e dopo lo sviluppo dei primi browser, è stato chiaro che occorreva aggiungere capacità di elaborazione alle pagine HTML
- **Client-Side Script**: programmi annegati nella pagina HTML che vengono interpretati all'interno del browser

JavScript

- Per questo è stato creato JavaScript
- Un programma JavaScript è in grado di interagire con la pagina, modificandola
- JavaScript è in qualche maniera ispirato a Java, ma è molto più semplice e adotta un diverso approccio alla programmazione a oggetti

Caratteristiche

- JavaScript fa una valutazione dinamica
  - delle istruzioni
  - dei tipi delle variabili
- **Vantaggi**: si può programmare in modo veloce e flessibile se la complessità è ridotta
- **Svantaggi**. difficoltà di debugging elevata

JavaScript e HTML

- Un elemento specifico racchiude tutto il codice JavaScript

```
<script type="text/javascript">
  ...
</script>
```

Che può essere ovunque nel testo e ripetuto

JavaScript e HTML

Ma di versioni di JavaScript ve ne sono diverse. L’attributo **`language`** consente di specificare la versione

JavaScript e HTML

```
<script type="text/javascript"
        language="javaScript1.5">

  ...
</script>
```

Lo script viene eseguito se la versione del linguaggio è quella specificata

JavaScript e HTML

Ma è meglio non essere troppo specifici

```
<script language="javaScript">
  ...
</script>
```

JavaScript e HTML

Oppure non dirlo affatto, perché JavaScript è il linguaggio di default

```
<script>
  ...
</script>
```

Elementi di Base

Variabili

- Definizione:

```
var v = 1
var n = "Pippo"
```

Variabili

- Il tipo di dato è derivato dinamicamente
- Esempi:

```
var v = 1          numero intero
var n = "Pippo"    stringa
```

Variabili e contesti

- Facendo alcune prove, si è indotti a pensare che la parola chiave **`var`** non serva.
- Perché all'apparenza funziona tutto nello stesso modo
- Non è così: **`var`** contestualizza la variabile localmente (vedremo di più dopo)

Assegnamento

- Operatore: = (singolo, perché == confronta)

```
A = 2;
B = A * 3;
```

I punti e virgola?

- In JavaScript, il `;` è un separatore di istruzioni sulla stessa riga
- Però, non è una brutta prassi di programmazione metterlo comunque, aiuta a prevenire errori

Operatori Aritmetici

- `a + b` (somma)
- `a - b` (sottrazione)
- `a * b` (prodotto)
- `a / b` (divisione)

Assegnamenti Incrementali

- `a = a + b` (somma)
- `a = a - b` (sottrazione)
- `a = a * b` (prodotto)
- `a = a / b` (divisione)

Assegnamenti Incrementali

- `a += b` (somma)
- `a -= b` (sottrazione)
- `a *= b` (prodotto)
- `a /= b` (divisione)

Operatori di Incremento

Prima valuta, poi incrementa

- `a++` (incremento di uno)
- `a--` (decremento di uno)

Prima incrementa, poi valuta

- `++a` (incremento di uno)
- `--a` (decremento di uno)

Istruzioni di controllo

- Come in C, C++, Java
- if
- while
- for

Operatori di confronto

- `A > 0`
- `A >= 0`
- `A < 0`
- `A <=0`
- `A == 0`
- `A != 0`

Operatori Logici

- `&&` AND
- `||` OR
- `!` NOT

Funzioni

Sintassi:

```
function nome(  parametri formali )
{
   codice
}
```

Esempio: somma

```
function somma(x, y)
{
    var r = x + y;
    return r;
}
```

Valutazione Dinamica dei Tipi

- Il tipo dei parametri formali e del valore restituito dalla funzione non sono specificabili
- Vengono valutati dinamicamente

Funzioni: Uso

```
var A;
var B;
var  C;


A = 2;
B = A * 3;
C = somma(A, B);
```

Valore Calcolato:
8

Funzioni: Uso

```
var A;
var B;
var  C;


A = "Salve ";
B = "Mondo";
C = somma(A, B);
```

Valore
Calcolato:
"Salve Mondo"

Funzioni senza Nome

- Le funzioni possono essere definite senza nome

```
function (x, y)
{
    var r = x + y;
    return r;
}
```

Funzioni senza Nome

- Perché sono assimilate agli oggetti
- Per esempio, possono essere assegnate a variabili

```
var fsomma = function (x, y)
{
    var r = x + y;
    return r;
}
```

Chiamata di funzione da variabile

```
var A;
var B;
var  C;


A = 2;
B = A * 3;
C = fsomma(A, B);
```

Valore
Calcolato:
8

Oggetti

Oggetti

- In JavaScript, un oggetto è un dato complesso
- Cioè può avere dei campi, ognuno con un nome diverso
- Anche il concetto di metodo è supportato
- È a metà tra le strutture del C e gli ogetti di Java

Oggetti

- Il modello degli oggetti di JavaScript è «molto particolare»
- Ha delle implicazioni inaspettate, quando si inizia a usare JavaScript
- Esistono tre modalità per definire oggetti inJavaScript

Modalità 1: Estensione di Object

```
pers = new Object()
pers.name = "Giuseppe"
pers.height = "1.80m"
```

- Object è l'oggetto base, vuoto (o quasi)
- I campi **`name`** e **`height`** vengono aggiunti dopo, ma poi si possono usare (sempre con la notazione puntata)

Modalità 1: Estensione di Object

- I campi sono come delle variabili, solo che sono contenuti all’interno degli oggetti
- Allora, come le variabili, ai campi si può assegnare una funzione (METODO)

Modalità 1: Estensione di Object

```
pers.run = function()
 { this.state = "running"
   this.speed = "4m/s"
 }
```

`this` accede all'oggetto che contiene il metodo

Modalità 1: Definizione Completa

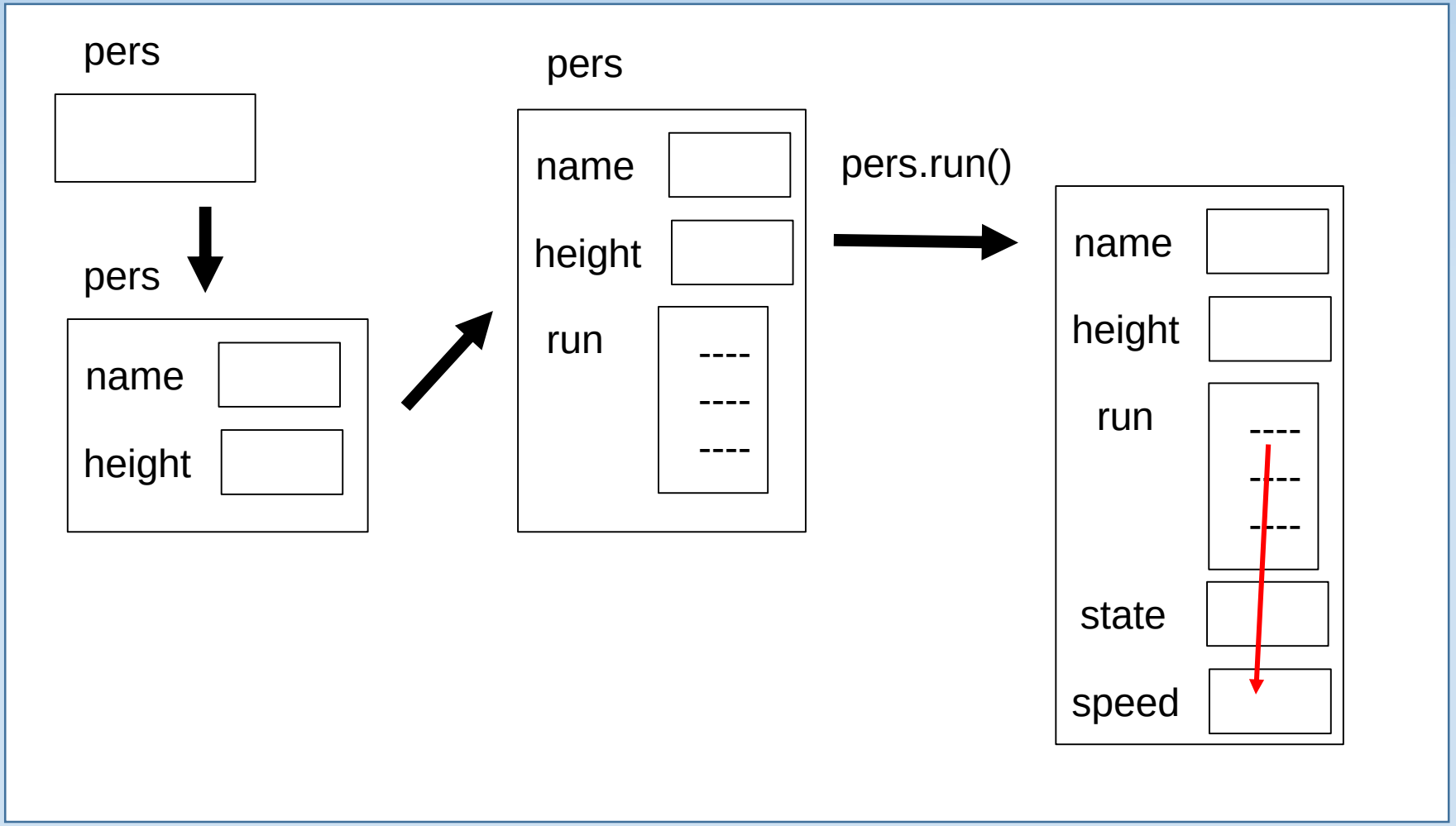
```
pers = new Object()
pers.name = "Giuseppe"
pers.height = "1.80m"
pers.run = function()
 { this.state = "running"
   this.speed = "4m/s"
 }
```

Modalità 1: Chiamata del Metodo

```
pers.run();
```

Dopo questa chiamata, l'oggetto **`pers`** ha due campi in più

Modalità 1


pers
pers
name
height
pers
name
height
run

----
pers.run()
name
height
run

----
state
speed

Modalità 2: Quasi una classe

- Creare una Struttura Comune da replicare facilmente.
- Nei linguaggi tipizzati viene chiamata Classe
- Javascript procede per Clonazione
- «Costruttore» di Oggetti:
  - Una funzione che inizializza le proprietà e i metodi di un oggetto in modo parametrico.

Modalità 2: «Costruttore»

```
function Person(pname, pheight) {
 this.name = pname
 this.height = pheight
 this.run = function()
 { this.state = "running"
   this.speed = "4m/s"
 }
}
```

Modalità 2: «Costruttore»

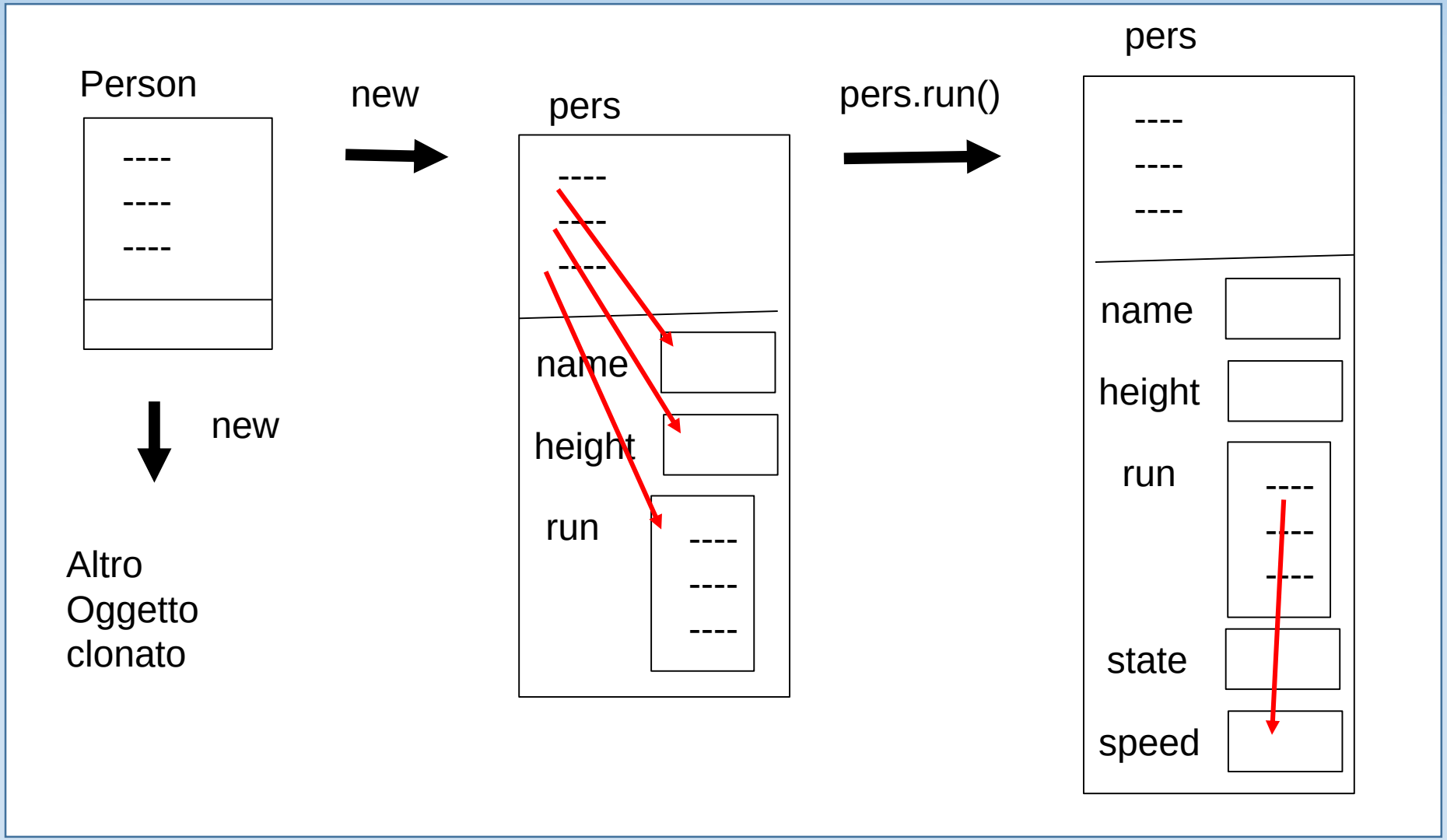
- Perché **`this`** nella funzione?
- Perché in realtà, un oggetto e una funzione **sono la stessa cosa**
- Sono due lati della stessa medaglia:
  - Il codice della funzione
  - Lo spazio dei dati
- **`this`** accede allo spazio dati dell'oggetto implicitamente associato alla funzione

Modalità 2: Clonazione

- **`new`** clona **`Person`**

```
var pers =
 new Person("Giuseppe", "1.80m" );
pers.run();
var pers2 =
 new Person("Paola", "1.60m" );
pers2.run();
```

Modalità 2


Person

----
new
Altro
Oggetto
clonato
new
pers

----
name
height
run

----
pers.run()
pers

----
name
height
run

----
state
speed

Modalità 2

- Quindi, la funzione clonata ha il compito di strutturare tutti gli oggetti derivati nello stesso modo
- E’ quasi una classe
- Una volta che l’oggetto è stato clonato, la sua struttura può essere estesa ulteriormente

Modalità 3

- Oggetti come Literals:
- All’interno di parentesi graffe
  si definiscono al volo i campi e i metodi
- Pratico per cose semplici o uniche
  non consente il riuso

Modalità 3

```
pers = {
 name : "Giuseppe",
 height : "1.80m",
 run : function()
 { this.state = "running"
   this.speed = "4m/s" }
}
```

Modalità 3

- Anche gli oggetti creati con la modalità 3 possono essere ulteriormente estesi
- Basta definire un nuovo campo.
- Sono molto utili per predisporre degli oggetti di configurazione, normalmente richiesti dai framework.

Information Hiding?

- JavaScript non supporta l'Information hiding
- Chiunque può vedere il contenuto di qualsiasi oggetto
- Gli oggetti sono completamente aperti

Oggetti Standard

Il linguaggio fornisce due tipi di oggetti standard

- Array
- String

Array

- `var a = new Array(10)`

- L’array è un oggetto
- Il valore tra parentesi è il numero di elementi iniziali
- Gli elementi dell’array non sono tipizzati
- Uso:
  `a[0] = "Pippo"; a[1] = 2`

Array

Proprietà:

- `length` **elementi nell’array**

Array

Metodi (alcuni)

- **`shift()`** Restituisce il primo valore e lo toglie
- **`unshift(v)`** Inserisce in cima il valore **`v`**
- **`push(v)`** accoda **`v`**
- **`pop()`** Restituisce e rimuove l'ultimo valore

Array

Metodi (alcuni)

- **`indexOf(v)`** Restituisce la posizione di v (-1 se non è presente)

Stringhe

- Oggetti di tipo **`String`**
- Le costanti stringa `"testo"` o `'testo'` vengono automaticamente convertite nell'oggetto corrispondente
- La «classe» **`String`** fornisce molti metodi per manipolare le stringhe, molti di questi pensati per il Web

Stringhe

- **String: Campi**

`length` **Numero di caratteri della stringa**

Stringhe

- String: Metodi

String anchor(name)
Genera la stringa con l'elemento
<A name="name">

String big()
Inserisce la stringa in <BIG>

String blink()
Inserisce la stringa in <BLINK>

String bold()
Inserisce la stringa in <B>

Stringhe

String fixed()
Inserisce la stringa in <TT>

String fontcolor(color)
Inserisce la stringa in
<FONT color="color">

String fontsize(size)
Inserisce la stringa in <FONT size="size">

String italics()
Inserisce la stringa in <I>

Stringhe

String link(url)
Inserisce la stringa in <A href="url">

String small()
Inserisce la stringa in <SMALL>

String strike()
Inserisce la stringa in <STRIKE>

String sub()
Inserisce la stringa in <SUB>

Stringhe

String sup()

Inserisce la stringa in <SUP>

String toLowerCase()

Ritorna la stringa con tutti i caratteri convertiti in minuscolo.

String toUpperCase()

Ritorna la stringa con tyutti i caratteri convertiti in masiuscolo.

Stringhe

String charAt(pos)

Restituisce il carattere in posizione pos

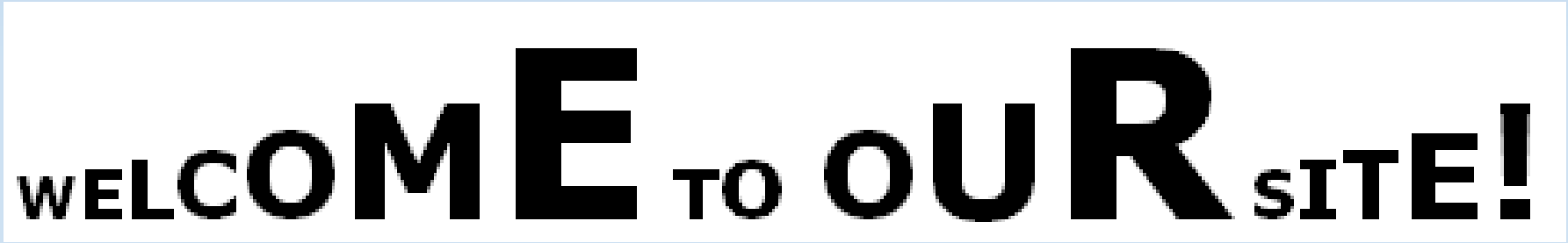
Stringhe: Esempio

```
var message="Welcome to our site!"
var format=message.toUpperCase()
var size=1 //go through the message,
 letter by letter
for (i=0;i<format.length;i++){
 document.write(
   format.charAt(i).fontsize(size).bold())
 if (size<7) size++; else size=1; }
```

Stringhe: Esempio

```
<B><FONT size="1">W</FONT></B>
<B><FONT size="2">E</FONT></B>
<B><FONT size="3">L</FONT></B>
...
```

Stringhe: Esempio



const, var, let

- const: la variabile non è modificabile (è una costante)
  const d=1;
- var: contestualizza la variabile nella funzione o nell’oggetto di contesto
  var d=1;
  d=2;

const, var, let

- let: contestualizza la variabile nel blocco di codice che contiene la definizione (per esempio, tra graffe)

```
var d=1;
{ let d=2; }
// qui d vale 1
```