

# LINGUAGGI PER IL WEB CLIENT-SIDE

---

*Engim Torino - Tecnico sviluppo software*





# LINGUAGGI PER IL WEB CLIENT-SIDE

---

*Engim Torino - Tecnico sviluppo software*



# PREREQUISITI

---

*Engim Torino - Tecnico sviluppo software*

# PREREQUISITI

---

*Sviluppo Software **VS** Programmazione*



# PREREQUISITI

---

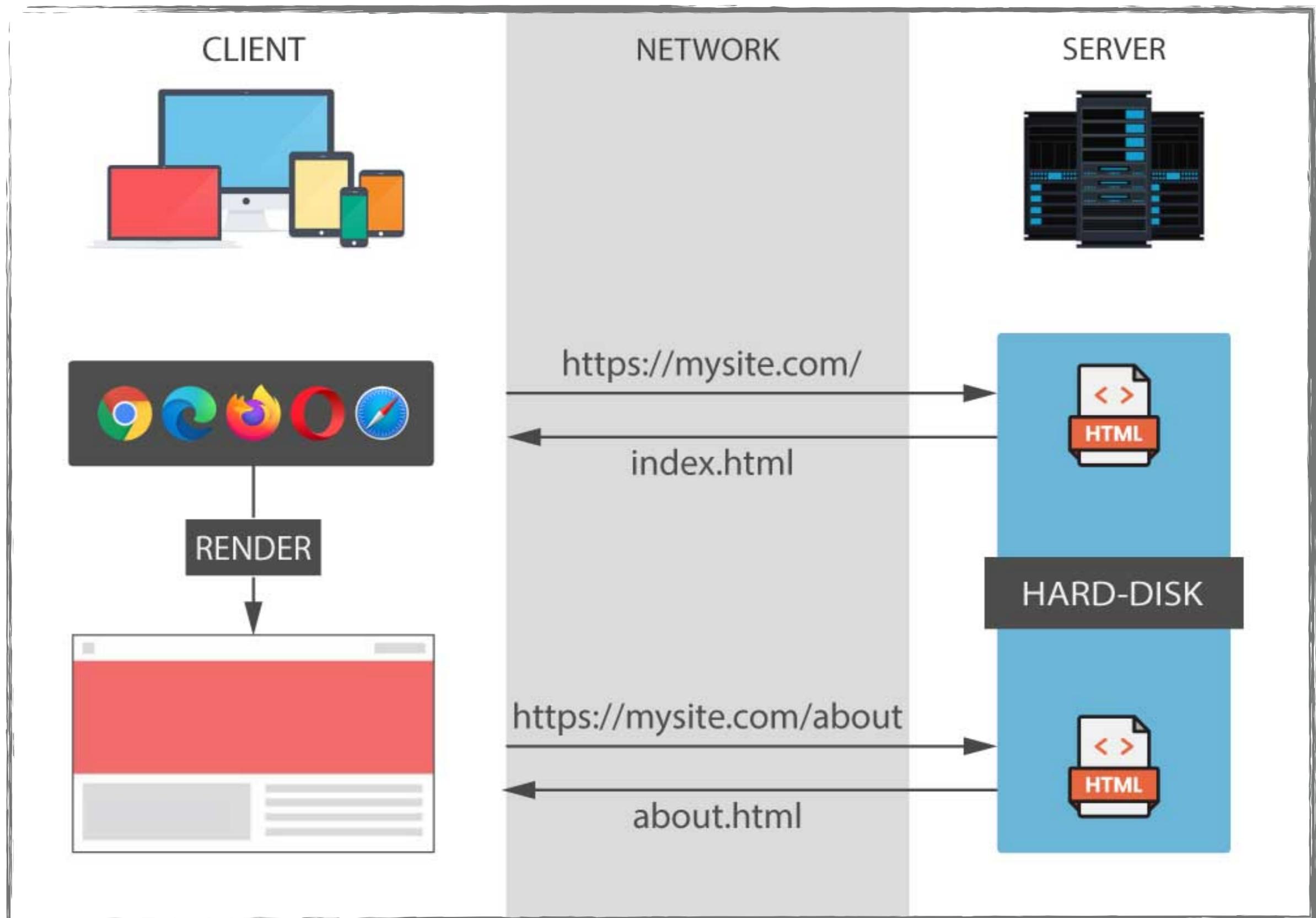
*Procedura **VS** Algoritmo*



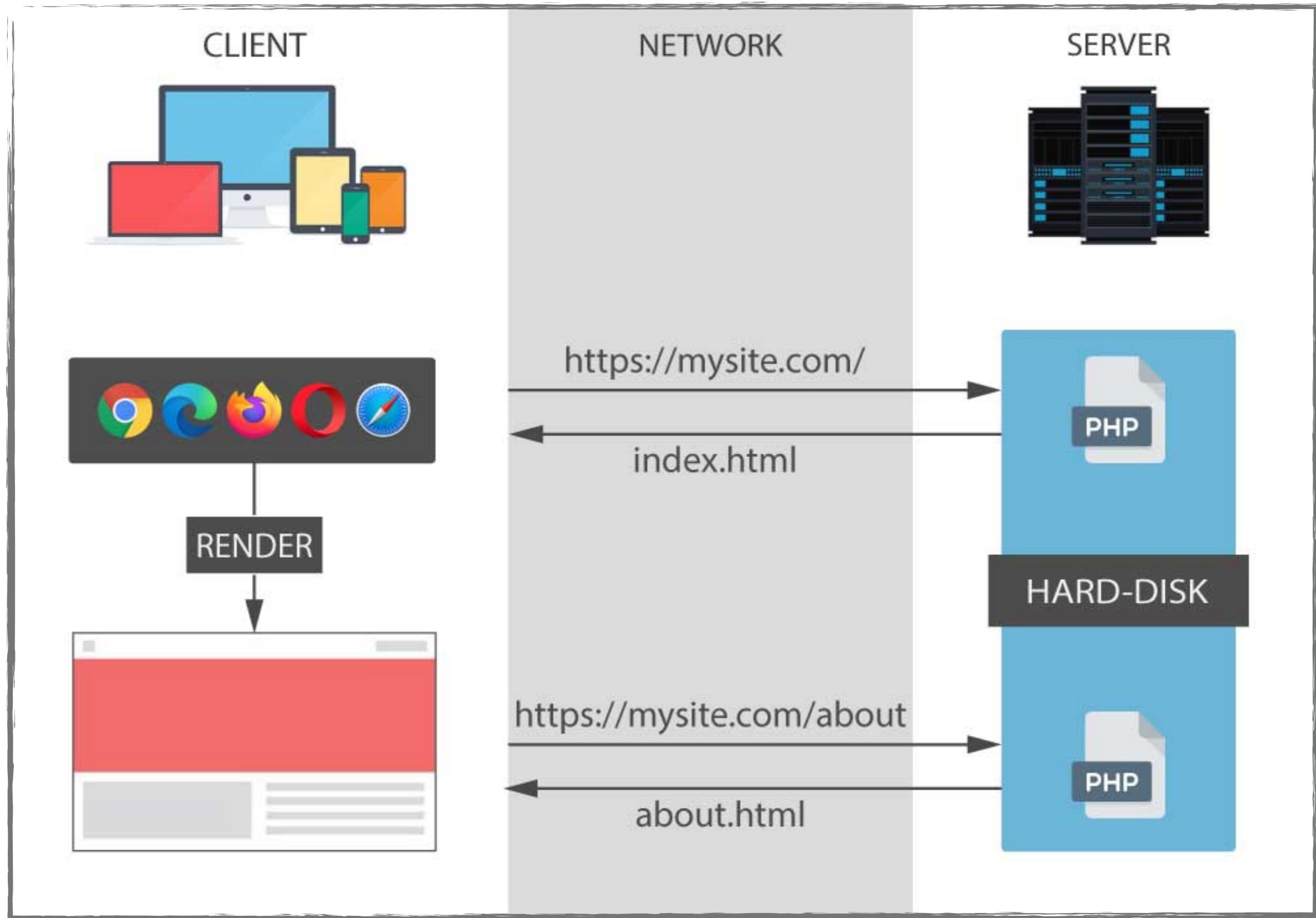


VIAGGIO NEL TEMPO

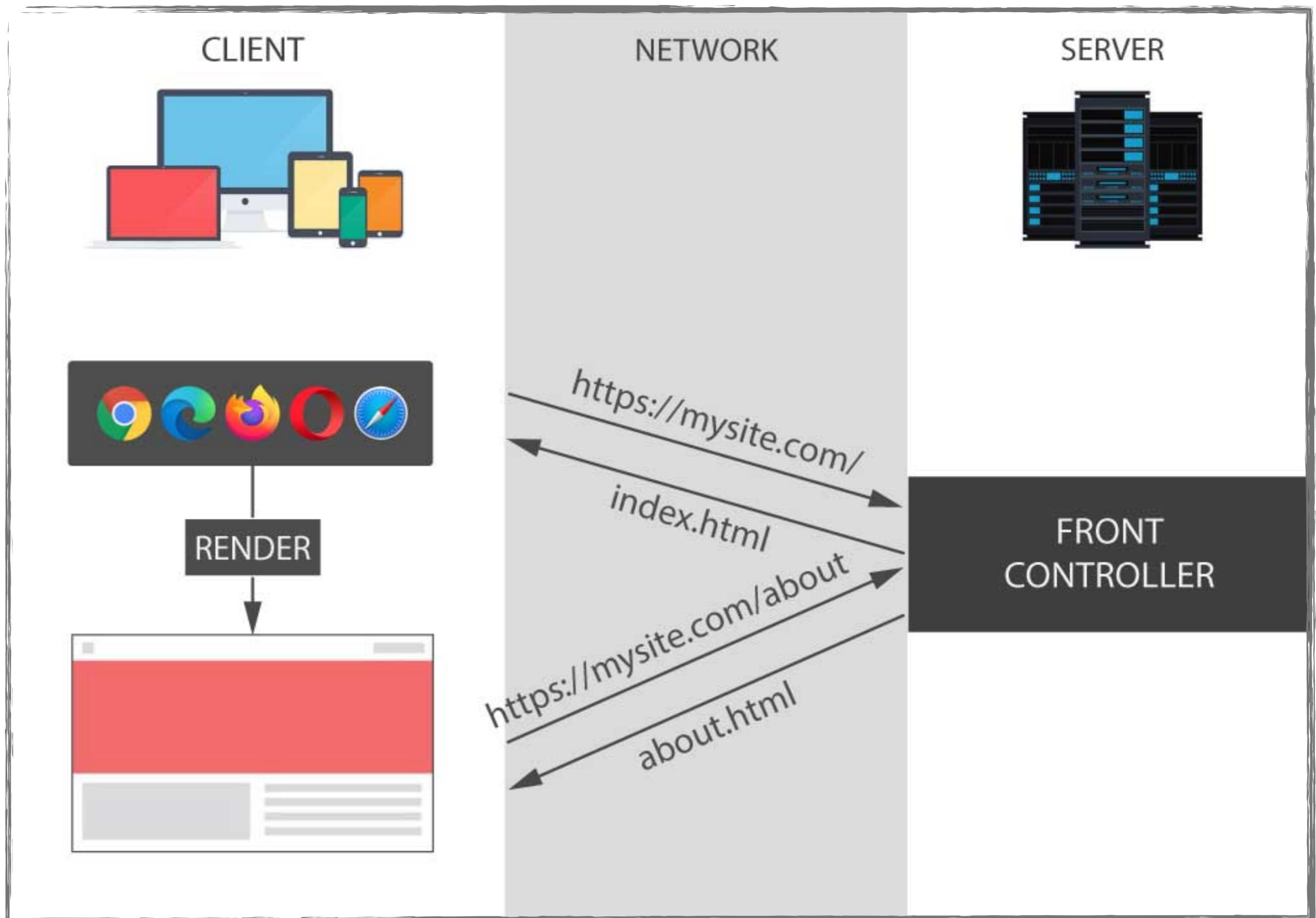
# STATIC SITES - 1995+



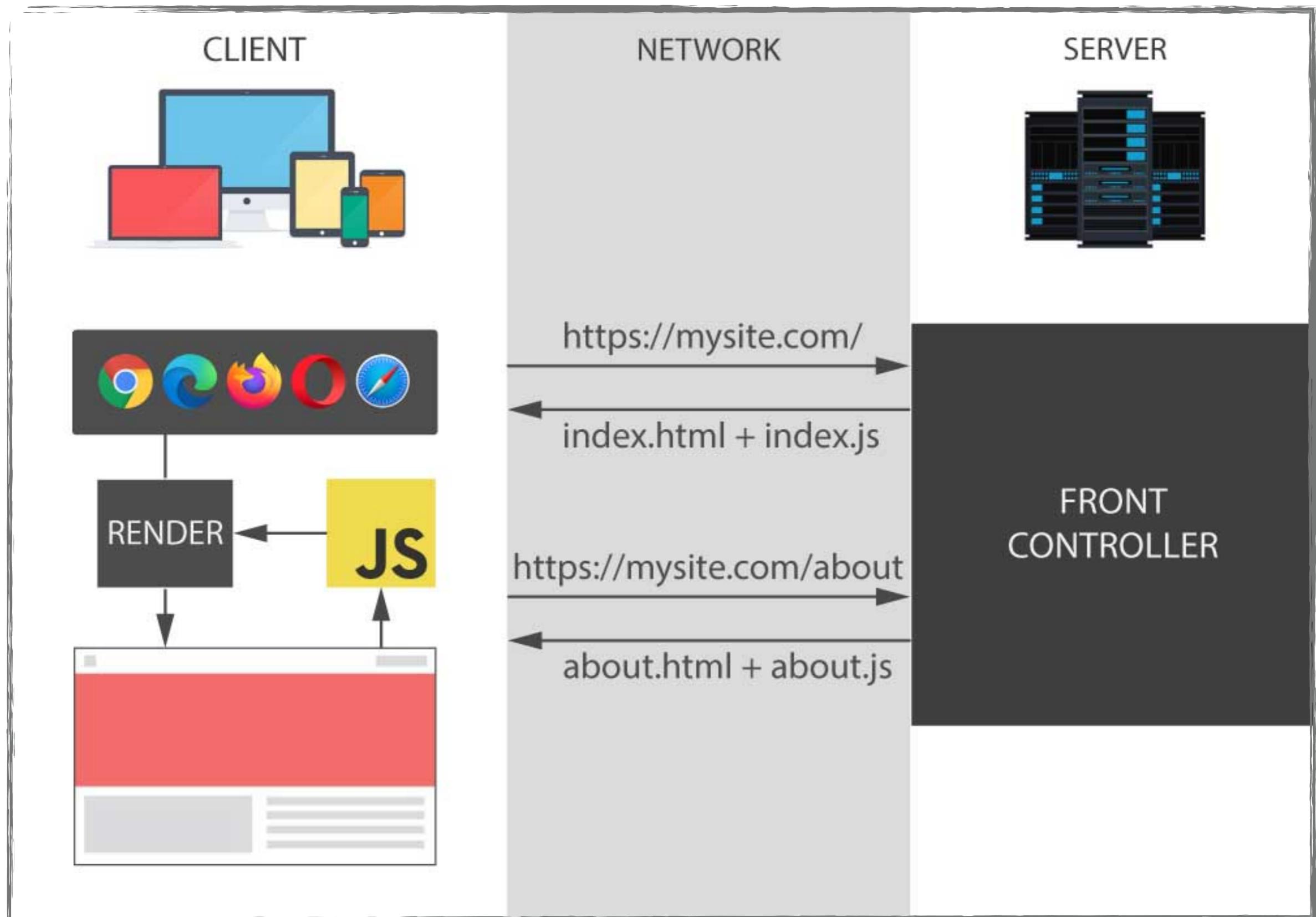
# SERVER SIDE - 2000+



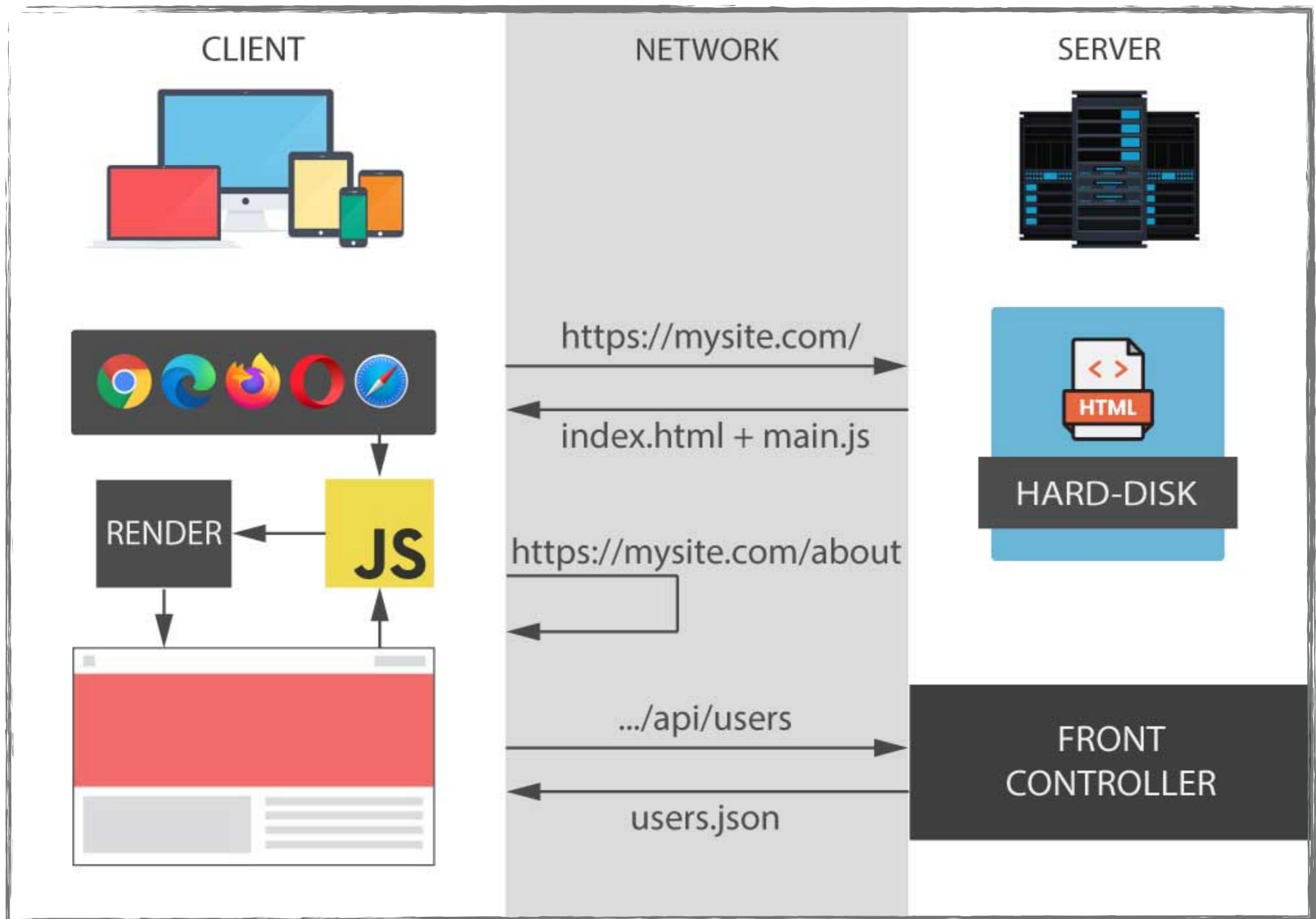
# MVC FRAMEWORKS - 2005+



# JAVASCRIPT - 2005+



# SINGLE PAGE APPLICATION(SPA) - 2015+





JAVASCRIPT

# SETUP IN-LINE

---

*Javascript dentro il tag <script></script>*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <script>
    console.log('pippo')
  </script>
</body>
</html>
```

*Poco adatto ad applicazioni complesse*

Firefox File Edit View History Bookmarks Tools Window Help

Lacentralina - Home Page https://futureintheair.net

Downloads ⌘ J Add-ons and Themes ⌘ A Sign In

Browser Tools ⌘ I Page Info ⌘ U

Web Developer Tools

- Task Manager
- Remote Debugging
- Browser Console ⌘ J
- Responsive Design Mode ⌘ M
- Eyedropper
- Page Source ⌘ U
- Extensions for Developers

1

# What you get with aGrid

Environmental data is important to understand the place where you live and where you practice your daily activities

## Features

- Real time information
- High level data quality
- Up to 400m resolution
- Pan-european area coverage
- Direct access to your own monitoring device data
- Personalised alerts
- Monitoring device information sharing
- Patented estimate of your physiological pollutant absorption

Explore the air pollution data anywhere in Europe, find out how it impacts your health.

\* Generated using Copernicus Atmosphere Monitoring Service

Add Your Lacentralina Device

You can add your Lacentralina device to see the data it collects.

Inspector Console Debugger Style Editor Performance Memory Network Storage Accessibility Application

Filter Output

This page uses the non-standard property zoom. Consider using calc() in the relevant property values, or using "transform" along with "transform-origin: 0 0".

#FUTUREINTHEAIR VERSION: 19 novembre 2021 14:31

2

3

# SETUP - FILE DEDICATI

*Javascript in uno o più file separati*



*I file vengono eseguiti in ordine (dall'alto verso il basso)*  
Conteso globale

gattini - Ricerca Google

https://www.google.com/search?q=gattini&tbo=isch&ved=2ahUKEwi\_usvFyOX0AhWs\_rslHVYnAFoQ2-cCe...

images



gattini



Tutti

Immagini

Video

Shopping

Notizie

Altro

Strumenti

Inspector Console Debugger Style Editor Performance Memory Network Storage Accessibility Application

Filter URLs

| Status | Method | Domain                      | File   | Initiator          | Type | Transferred |
|--------|--------|-----------------------------|--|--------------------|------|-------------|
| 204    | POST   | www.google.com              | gen_204?atyp=i&ct=slh&cad=&v=t1&ei=ZL-5Yb_4PKz97_UP1s6A0AU&m=f                   | beacon             | html | 357 B       |
| 200    | GET    | www.google.com              | search?q=gattini&tbo=isch&ved=2ahUKEwi_usvFyOX0AhWs_rslHVYnAFoC m=ws9Tlc,IZT6... | m=ws9Tlc,IZT6...   | html | 407.45 kB   |
| 200    | GET    | www.google.com              | seasonal-holidays-2021-6753651837109324-s.png                                    | img                | png  | cached      |
| 200    | GET    | lh3.googleusercontent.com   | ADea4l4b9eiJCOoT0BbWnuuqSqlBeu06Gggb96Qq2_CC=s64-c-mo                            | img                | png  | cached      |
| 200    | GET    | www.gstatic.com             | photo_camera_gm_grey500_24dp.png   | img                | png  | cached      |
| 200    | GET    | www.gstatic.com             | m=_b,_tp   | script             | js   | cached      |
| 200    | GET    | ssl.gstatic.com             | arrow_down.png   | img                | png  | cached      |
| 200    | GET    | www.gstatic.com             | rs=AA2YrTsYklP8lcczE-ijd6RDYU_tD3G0Ag  | search:277 (scr... | js   | cached      |
| 204    | POST   | www.google.com              | gen_204?s=images_vfe&t=aft&atyp=csi&ei=er-5Yd7pMNmT7_UPz-OlsAc&i                 | search:7 (beac...  | html | 357 B       |
| 200    | GET    | www.gstatic.com             | m=byfTOb,lsjVmc,LEikZe   | m=_b,_tp:1493...   | js   | cached      |
| 200    | GET    | www.gstatic.com             | m=ws9Tlc,IZT63,n73qwf,UUJqVe,O1Gjze,xUdipf,blwjVc,fKUV3e,aurFic,COQ              | m=_b,_tp:1493...   | js   | 320.86 kB   |
| 200    | GET    | apis.google.com             | cb=gapi.loaded_0   | rs=AA2YrTsYkl...   | js   | cached      |
| 200    | GET    | www.gstatic.com             | m=llJq7e,nabPbb,TZG3Xc,z2BPKb,ANyn1,ONxwXc,t0CgGe,GFartf,LvPQXe,k                | m=_b,_tp:1493...   | js   | 39.94 kB    |
| 200    | GET    | www.gstatic.com             | m=XJI8jf   | m=_b,_tp:1493...   | js   | 1.43 kB     |
| 200    | GET    | www.gstatic.com             | m=JNcJEf,XXjTHd,Dverrd   | m=_b,_tp:1493...   | js   | 3.10 kB     |
| 200    | GET    | www.google.com              | search?q&cp=0&client=img&xssi=t&gs_ri=gws-wiz-img&ds=i&hl=it&authus              | m=_b,_tp:974 (...  | json | 1.09 kB     |
| 200    | GET    | www.gstatic.com             | m=sOXFj,lps5vc   | m=_b,_tp:1493...   | js   | 32.73 kB    |
| 200    | GET    | encrypted-tbn0.gstatic.c... | images?q=tbn:ANd9GcSk_TXTVkjbeDW-WXrjT87Lc4cR9dru44oQrXxIBKf                     | m=ws9Tlc,IZT6...   | jpeg | 4.08 kB     |
| 200    | GET    | encrvoted-tbn0.gstatic.c... | imaes?a=tbn:ANd9GcSzehpXMUuc0OcO8skJQ3tasrEkvRQGeXEDr88V6z                       | m=ws9Tlc,IZT6...   | ipea | 3.55 kB     |

56 requests 3.64 MB / 927.66 kB transferred Finish: 35.03 s DOMContentLoaded: 1.05 s load: 1.84 s

# TIPI DI DATO

---

*Number:* 42

*String:* “hello world”

*Boolean:* true

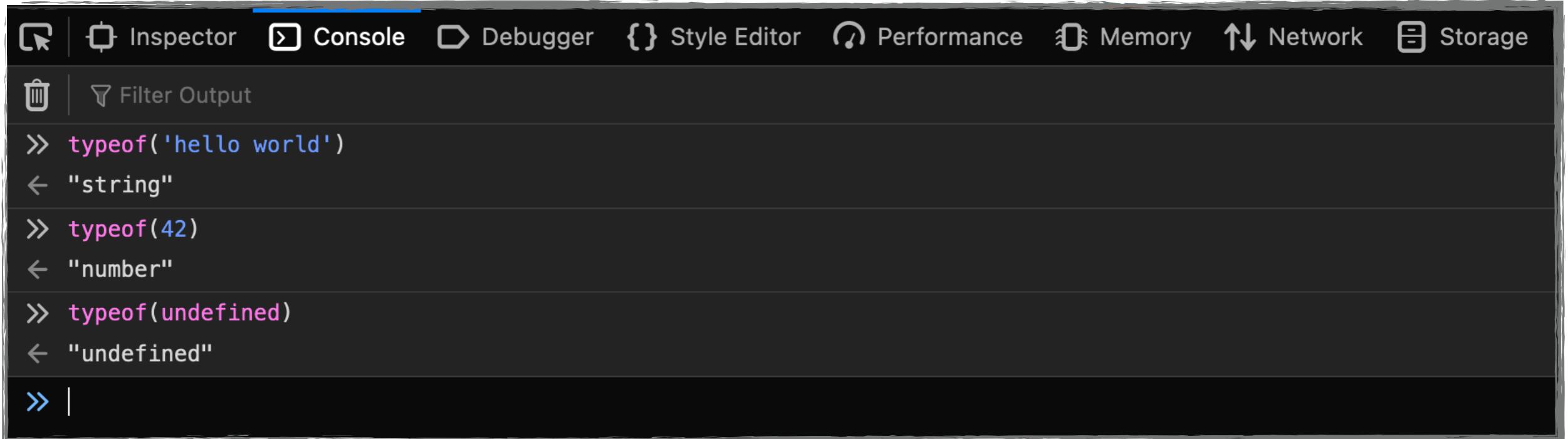
*Undefined:* undefined

*Null:* null



# OUTPUT TYPE

*typeof() ritorna una stringa contenente il NOME del tipo*



The screenshot shows a browser's developer tools interface with the "Console" tab selected. The console output displays the results of four typeof() calls:

```
» typeof('hello world')
← "string"
» typeof(42)
← "number"
» typeof(undefined)
← "undefined"
» |
```

# OPERATORI - ARITMETICI

```
// Addizione  
5 + 3  
// Sottrazione  
5 - 3  
// Moltiplicazione  
5 * 3  
// Divisione  
5 / 3  
// Resto della divisione  
5 % 3  
// Esponente  
5 ** 3  
  
// ESEMPIO  
(4 + 20) / 4 * 2
```



VARIABILI

# VARIABILE

*Nome arbitrario, contiene un singolo valore*

```
// Definizione + Assegnazione
let chooseAVariableName = "Some Value"
// Mostro il valore in console
console.log(chooseAVariableName)
```

```
// Modifica Del valore
chooseAVariableName = 50
// Mostro il valore in console
console.log(chooseAVariableName)
```

## **ATTENZIONE al NOME:**

*I caratteri consentiti sono: a-z A-Z 0-9 \$ \_*

*Il primo carattere non può essere un numero*

*Non puo essere uguale ad una parola riservata (ES: undefined, null, function)*

*Case sensitive*

# VARIABILI - ESEMPI DI OPERAZIONI

```
// ESEMPIO 1
```

```
let varA = 5
```

```
let varB = 10
```

```
let result = varA + varB
```

```
console.log(result)
```

```
//> 15
```

```
// ESEMPIO 2 (continuazione esempio 1)
```

```
varB = 14.5 + 0.5
```

```
let varC = varB / varA
```

```
result = result / (varC)
```

```
console.log(result)
```

```
//> 5
```

```
// ESEMPIO 3 (continuazione esempio 2)
```

```
console.log( (varA + varB + varC + 3.5) / result )
```

```
//> 5.3
```

# VARIABILI - COSTANTE

*Il valore non può essere modificato*

```
// Definizione + Assegnazione
const chooseAVariableName = "Some Value"

// Modifica Del valore
chooseAVariableName = 50    ERRORE: Assignment to constant variable
```

*Valgono le stesse regole delle variabili*

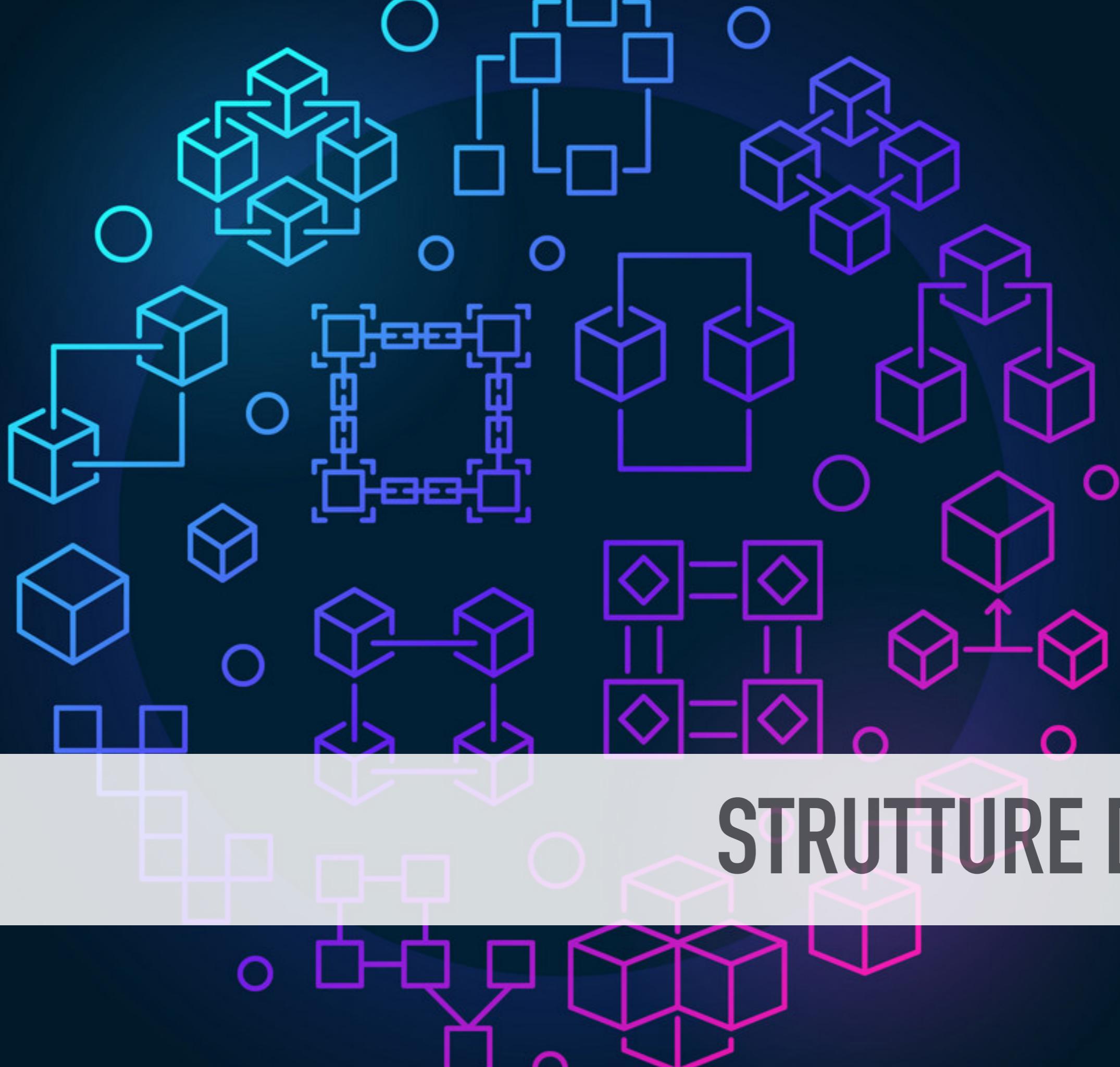
# VARIABILI - ASSEGNAZIONE

---

*Ad una variabile è possibile assegnare:*

- *Valori primitivi (Numeri, Stringhe, Valori Booleani, Null, Undefined)*
- *Espressioni*
- *Funzioni*
- *Strutture dati*

# STRUTTURE DATI



# STRUTTURE DATI

---

*Esistono diversi tipi di strutture dati con caratteristiche diverse tra loro*

*Servono a:*

- *Organizzare dati in gruppi*
- *Strutturare i dati in maniera conveniente*
- *Eseguire operazioni specifiche*

# ARRAY



# ARRAY



```
let a = ['aValue', 'b', 24, 'hola!', false, 'engim', null]
```

*Contiene valori in un ordine preciso*

*Ad ogni valore è associato un indice numerico univoco*

*I valori sono separati da una virgola*

## ATTENZIONE:

- *l'indice del primo elemento è 0*
- *Nelle versioni di javascript precedenti al 2015 non ammettono la virgola dopo l'ultimo elemento*

# ARRAY

```
// Definizione
let listName = [ 'aValue', 'aontherValue', 24, 'hola!' ]

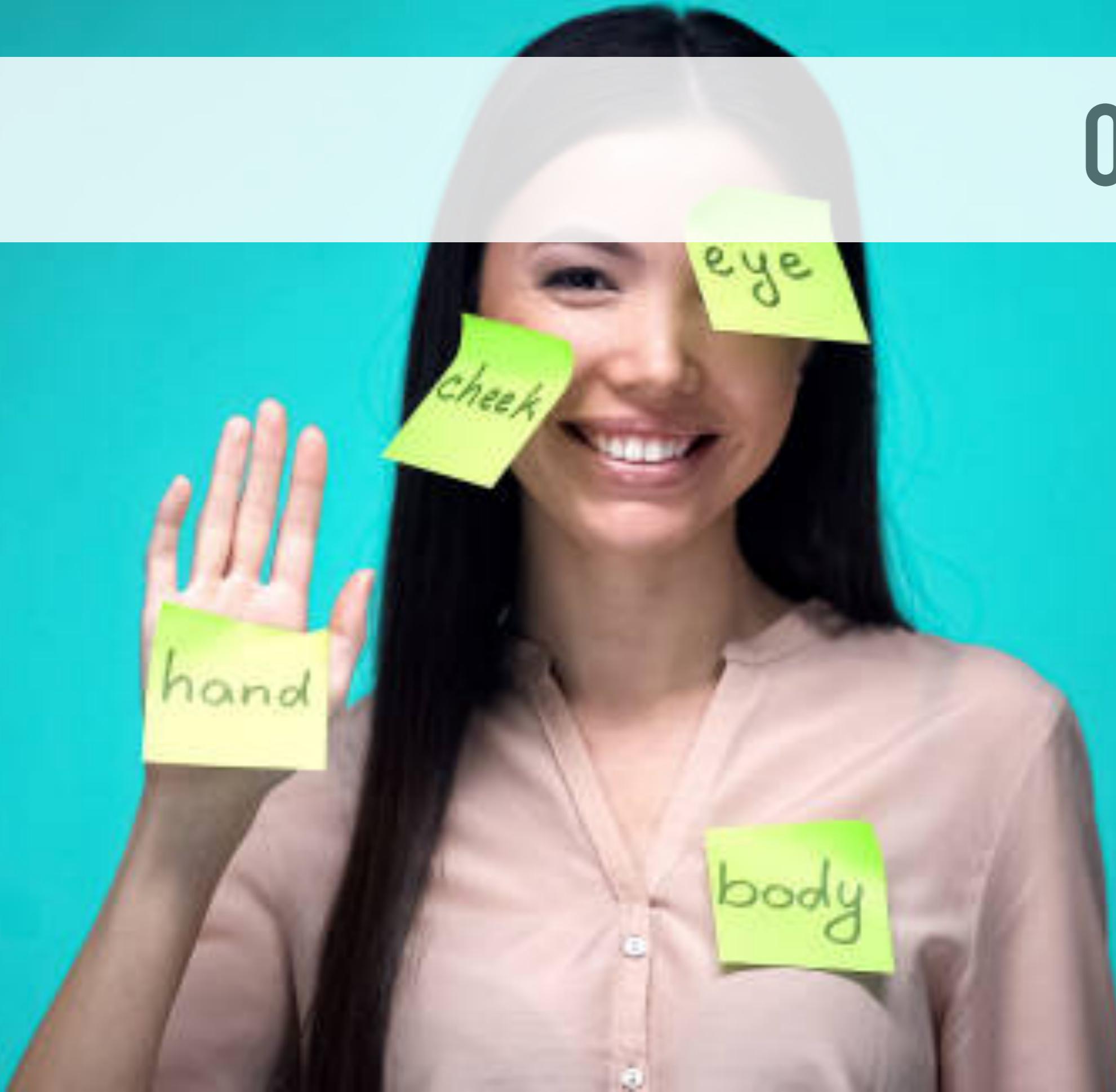
// Lettura
console.log(listName)
console.log(listName[1])

// Scrittura
listName[1] = 'changedValue'
listName[4] = ' newIndex'
console.log(listName)
```

*Si accede al valore tramite il suo indice*

*L'Array non ha una dimensione fissa*

# OGGETTI



# OGGETTO

*Collezione di elementi chiave-valore*

```
// Definizione
let objectName = {userName : 'admin', password : '12345678'}

// Lettura
console.log(objectName)
console.log(objectName.userName)

// Scrittura
objectName.password = 'qwerty'
objectName.email = 'admin@engim.com'
console.log(objectName)
```

**ATTENZIONE:** Gli elementi non hanno un ordine!

I nomi delle chiavi seguono le stesse regole dei nomi delle variabili

# OGGETTO - SINTASSI ALTERNATIVA

*Simile alla sintassi degli array*

```
// Definizione
let objectName = {"user name": 'admin', password: '12345678'}

// Lettura
console.log(objectName)
console.log(objectName['user name'])

// Scrittura
objectName['email'] = 'admin@engim.com'
console.log(objectName)
```

**UP:** con questa sintassi si possono aggirare le regole sui nomi delle chiavi

**UP:** si può anche usare una variabile contenente la chiave (stringa o numero)

# ESEMPI - OGGETTO - SINTASSI ALTERNATIVA

```
let persona = {  
    nome: 'pippo',  
    cognome: 'topolino',  
    "nome cane": "pluto",  
    "lavoro": 'detective'  
}  
  
/* Lettura */  
console.log(persona.nome) // Sintassi standard  
  
// E sempre possibile utilizzare la sintassi standard purchè  
// la chiave rispetti le regole dei nomi delle variabili  
console.log(persona.lavoro) // Sintassi standard  
  
  
console.log(persona['nome cane']) // Sintassi alternativa  
// La sintassi alternativa si può utilizzare sempre per accedere a qualsiasi  
elemento  
console.log(persona['nome']) // Sintassi alternativa  
  
/* Scrittura */  
persona['indirizzo'] = "via degli sviluppatori"
```

# ESEMPI - OGGETTO - SINTASSI ALTERNATIVA

---

Con la sintassi alternativa è possibile ottenere la chiave da una variabile che contenga una stringa. Valido sia in lettura che in scrittura.

```
// scrittura
let nomeChiave = 'eta'
persona[nomeChiave] = 25

// lettura
let x = 'nome'
console.log(persona[x])
```

# STRUTTURE DATI - ESEMPI

```
let recensori = [
  {
    id: 5,
    nome: "Yotobi",
    voti: [
      { id: 15, title: "Avengers Endgame", voto: 8},
      { id: 20, title: "Sharknado", voto: 10},
      { id: 36, title: "Joker", voto: 8},
      { id: 49, title: "In vacanza su marte", voto: 4},
    ]
  },
  {
    id: 55,
    nome: "Synergo",
    voti: [
      { id: 15, title: "Avengers Endgame", voto: 3},
      { id: 20, title: "Sharknado", voto: 6},
      { id: 36, title: "Joker", voto: 7},
      { id: 49, title: "In vacanza su marte", voto: 1},
      { id: 65, title: "1917", voto: 9},
      { id: 10, title: "L'esorciccio", voto: 9},
    ]
  },
  {
    id: 90,
    nome: "BarbascuraX",
    voti: [
      { id: 20, title: "Sharknado", voto: 6},
      { id: 108, title: "Kung Fu Panda", voto: 10},
    ]
  }
]
```



# STRUTTURE DATI - NEXT?

---

*Set: simili agli array ma senza possibilità di valori duplicati*

*Map: simili agli oggetti ma ordinati*

# OPERATORI - COMPARAZIONE

---

*Restituisce sempre un valore BOOLEAN*

```
5 == 3+2 // risultato: true  
5 != 3+2 // risultato: false
```

```
5 > 3+2 // risultato: false  
5 >= 3+2 // risultato: true
```

```
5 < 3+2 // risultato: false  
5 <= 3+2 // risultato: true
```

# OPERATORI - COMPARAZIONE

*L'operatore == prova ad eseguire una conversione di TYPE*

```
// EQUALITY (LOOSE COMPARISION)
5 == 3+2 // risultato: true
"3" == 3 // risultato: true
0 == false // risultato: true
1 == true // risultato: true
"" == false // risultato: true
```

```
// IDENTITA (STRICT COMPARISION)
5 === 3+2 // risultato: true
"3" === 3 // risultato: false
0 === false // risultato: false
1 === true // risultato: false
"" === false // risultato: false
```



# STRUTTURE DI CONTROLLO

---

*Costrutti condizionali*

Esegue una porzione di codice in maniera condizionale

```
let userAge = 18

if(userAge >= 18){
    console.log('sei maggiorenne')
}
```

# IF-ELSE

---

*Esegue blocco ELSE solo se il blocco IF non viene eseguito*

```
let age = 18

if(age >= 18){
    console.log('sei maggiorenne')
}
else {
    console.log('sei minorenne')
}
```

**ATTENZIONE:** Non è permesso inserire codice tra i blocchi IF ed ed ELSE (sono permessi solo commenti e spazi)

# IF | ELSE IF | ELSE

*Un blocco viene eseguito solo se nessuno dei blocchi precedenti è stato eseguito*

```
let age = 18

if(age === undefined){
    console.log('Età non specificata')
}
else if(age >= 18){
    console.log('Sei maggiorenne')
}
else {
    console.log('Sei minorenne')
}
```

*Si possono aggiungere più blocchi “ELSE IF” ma solo un “IF” ed un “ELSE”*

**ATTENZIONE:** L’ordine è determinante



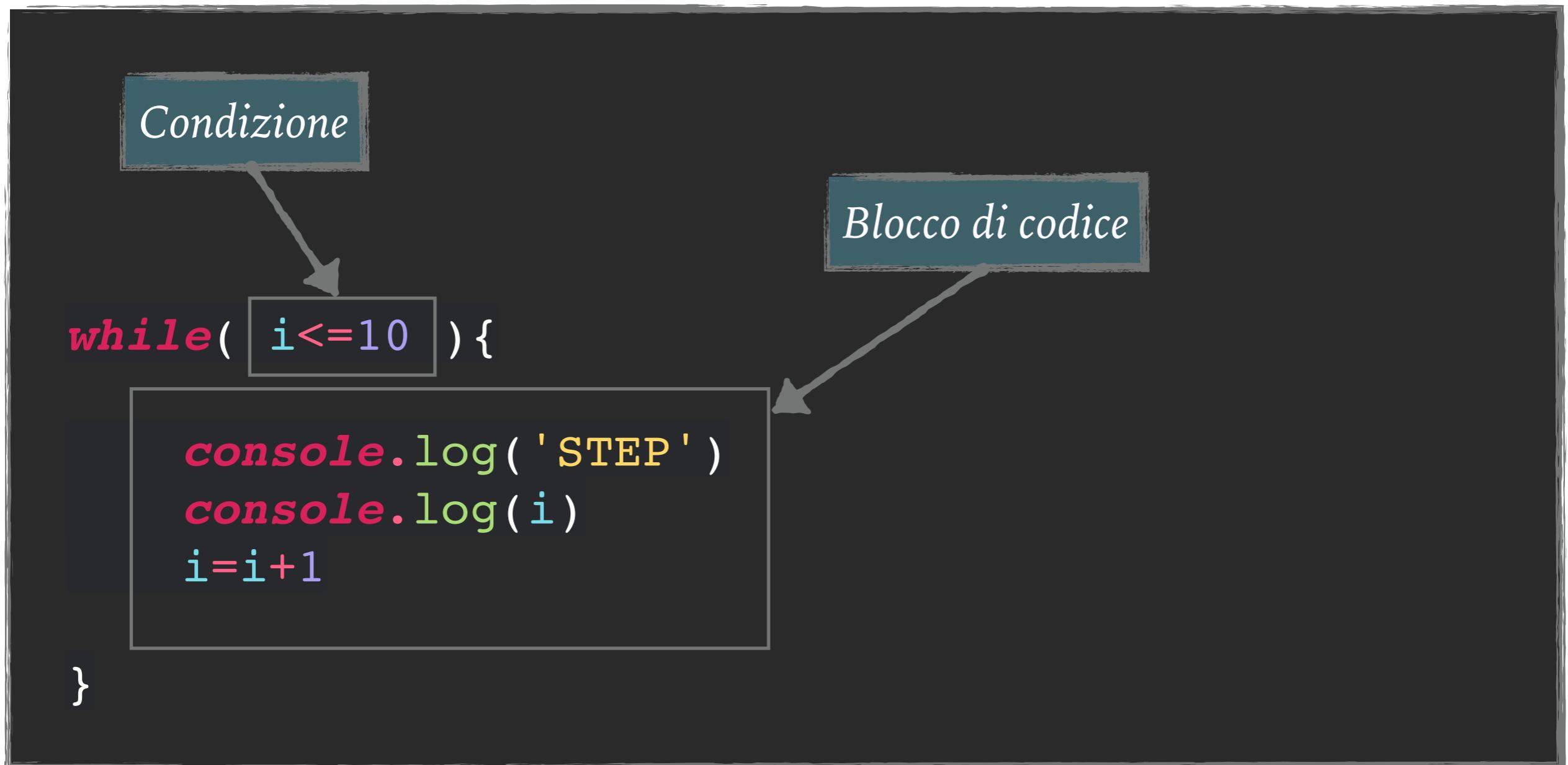
# STRUTTURE DI CONTROLLO

---

*Cicli ed Iterazioni*

# WHILE-LOOP

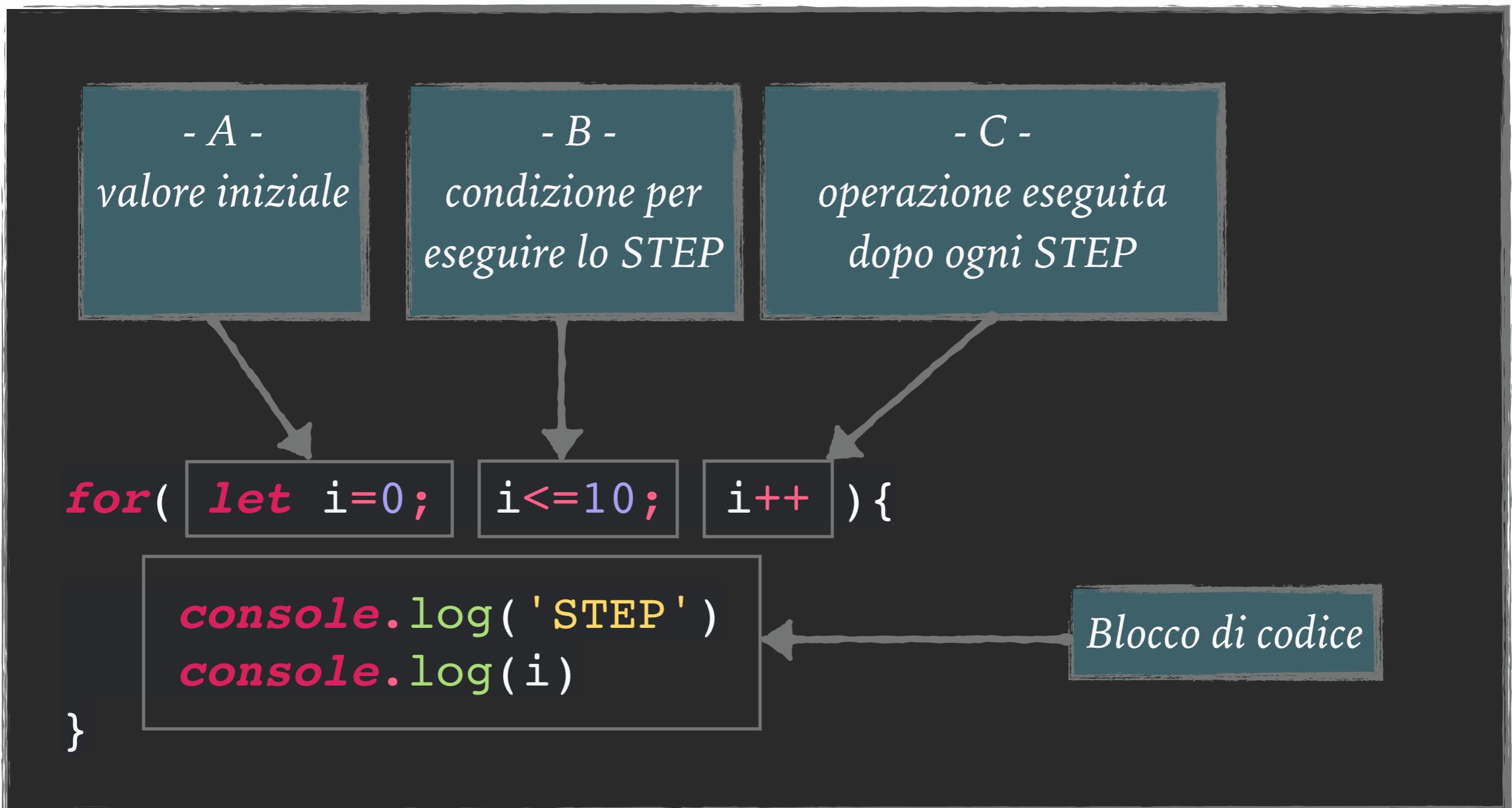
Esegue ripetutamente il blocco di codice fino a che la condizione non è FALSE



**ATTENZIONE:** assicurarsi di non creare un **INFINITE-LOOP**

# FOR-LOOP

*Simile al WHILE-LOOP*



# ESEMPI - ITERARE UN ARRAY - FOR-LOOP VS WHILE-LOOP

```
let lettere = ["alfa", "beta", "gamma", "delta", "epsilon", "zeta"]

// FOR-LOOP
for(let i=0; i<lettere.length; i++){
    console.log(lettere[i])
}

// WHILE-LOOP
let i=0;
while(i<lettere.length){
    console.log(lettere[i])
    i=i+1
}
```

# FOR...OF LOOP

*Specifico per iterare strutture dati ordinate (es: gli array)*

```
let lettere = ["alfa", "beta", "gamma", "delta", "epsilon", "zeta"]

for(let item of lettere){
    console.log(item)
}
```

*Non è possibile ottenere l'indice*

# FOR...IN LOOP

*Ad ogni ciclo i è l'indice*

```
let lettere = ["alfa", "beta", "gamma", "delta", "epsilon", "zeta"]

for(let i in lettere){
    console.log(lettere[i])
}
```

*L'unico ciclo che puo iterare anche strutture non ordinate (es: oggetti)*

**ATTENZIONE:** Iterare strutture non ordinate oggi è considerata *BAD-PRATICE*  
da molti sviluppatori

# CONTINUE

*Salta alla fine dello step corrente*

```
let lettere = ["alfa", "beta", "gamma", "delta", "epsilon", "zeta"]

for(let index=0; index<lettere.length; index++){
    // Eseguito ad ogni step
    console.log(index)

    // Salta allo step successivo se index non è dispari
    if(index%2 === 0){ continue }

    // Eseguito solo per indice dispari
    console.log(lettere[index])
}
```

**ATTENZIONE:** ignora il codice successivo al CONTINUE dello step corrente

# BREAK

*Interrompe l'intero ciclo*

```
let lettere = ["alfa", "beta", "gamma", "delta", "epsilon", "zeta"]

for(let index=0; index<lettere.length; index++){
    console.log(lettere[index])

    // interrompe il ciclo quando il valore è uguale a 'gamma'
    if(lettere[index]==='gamma') { break }

    console.log( '-')
}
```

**ATTENZIONE:** ignora il codice successivo al BREAK dello step corrente



# STRUTTURE DI CONTROLLO

---

*Funzioni*

# FUNZIONE

---

*Permette di dichiarare un blocco di codice ed eseguirlo successivamente*

*Una funzione può essere utilizzata in diverse parti della stessa applicazione*

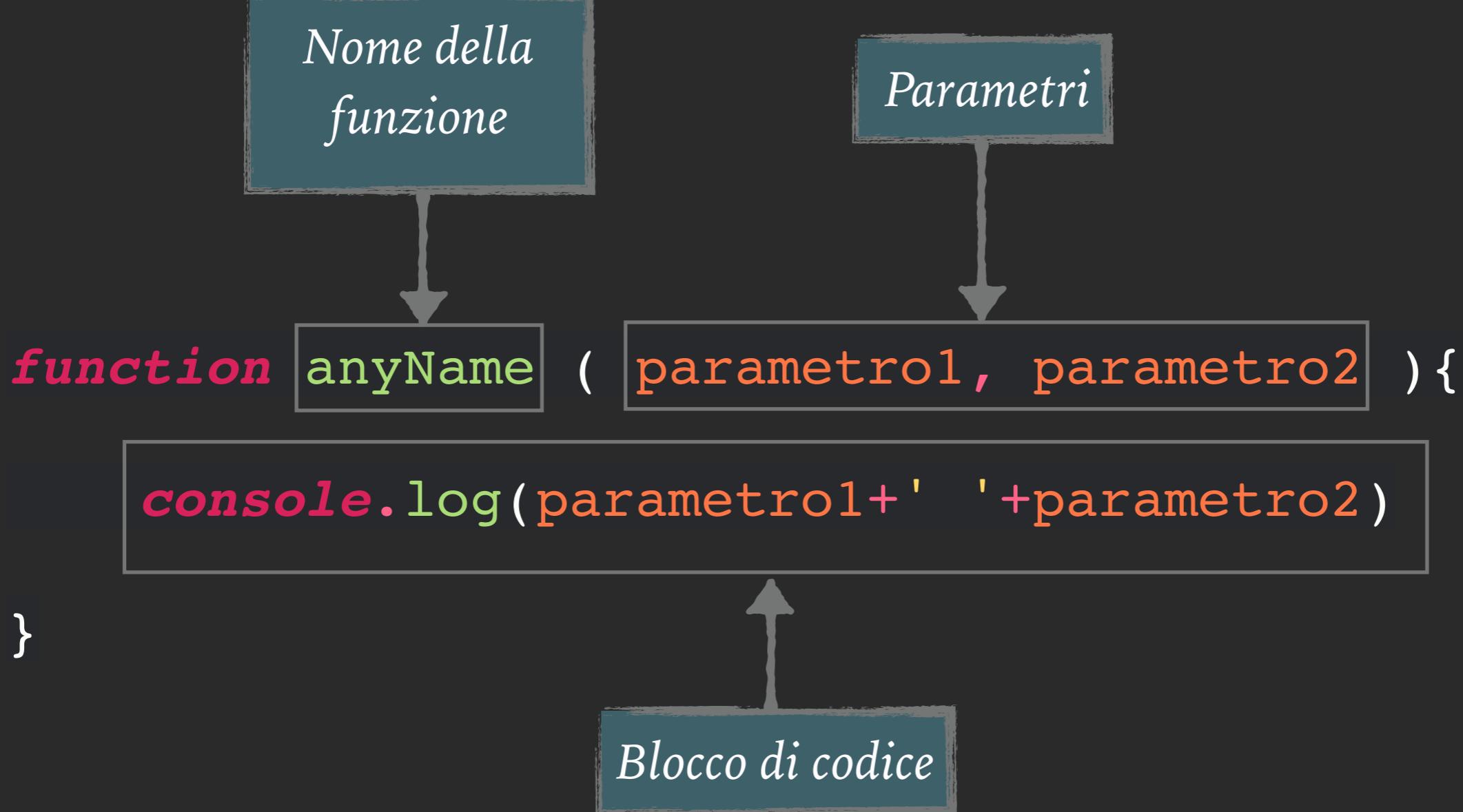
*Può accettare in input dei valori (Parametri)*

*Può produrre un risultato (Valore di ritorno)*

*Esistono funzioni predefinite (ES: typeof(), setTimeout ...)*

# FUNZIONE - SINTASSI

Dichiara un blocco di codice che puo essere eseguito successivamente



# ESEMPI - FUNZIONE SENZA VALORE DI RITORNO

```
// Dichiarazione
function checkPasswordValid(password){
    if(password.length < 8){
        console.log('Password non valida')
    }
    else{
        console.log('Password corretta')
    }
}

// Esecuzione (ES: Durante registrazione)
let userPassword = '12345678'
checkPasswordValid(userPassword)

// Esecuzione (ES: Durante cambio password)
let nuovaPassword = '1234'
checkPasswordValid(nuovaPassword)
```

**ATTENZIONE:** il nome del valore all interno della funzione è “password”

# ESEMPI - FUNZIONE CON VALORE DI RITORNO

```
// Dichiarazione
function isPasswordValid(password){
  if(password.length < 8){ return true }
  else{ return false }
}

// Esecuzione
let userPassword = '12345678'
let result = isPasswordValid(userPassword)
console.log(result)
```

# ESEMPI - FUNZIONE

```
function filterEvenElements(list){
  let results = []
  for(let index=0; index<list.length; index++){
    if(index%2 === 0){ results[results.length] = list[index] }
  }
  return results
}

// Esecuzione
let lettere = ["alfa", "beta", "gamma", "delta", "epsilon", "zeta"]
let lettereFiltered = filterEvenElements(lettere)
console.log(lettereFiltered)
// risultato: ['alfa', 'gamma', 'epsilon']

// Esecuzione
let mixed = ['spiderman', 'engim', 5, null, 3.5, 'hello', 'world' ]
let mixedFiltered = filterEvenElements(mixed)
console.log(mixedFiltered)
// risultato: ['spiderman', 5, 3.5, 'world']
```

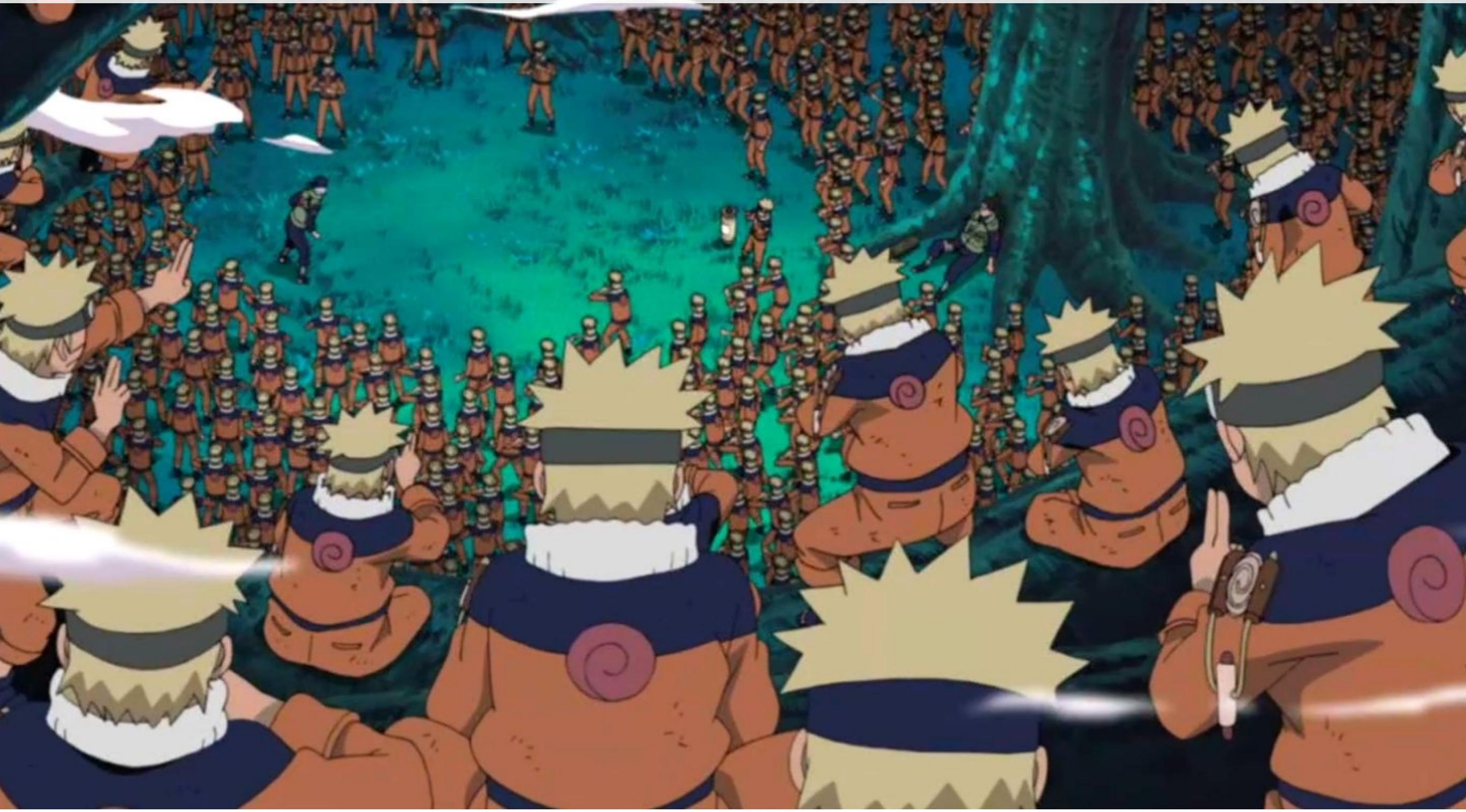
# ESEMPI - FUNZIONE SENZA INPUT, SENZA VALORE DI RITORNO

```
// Dichiarazione
function chiediNome(){
    let nome = prompt('Scrivi il tuo nome')
    let cognome = prompt('Scrivi il tuo cognome')

    alert('Ciao ' + nome + ' ' + cognome + ' !')
}

// Esecuzione
chiediNome()
```

# COPY VS REFERENCE



# COPY VS REFERENCE

---

*Copy:*

- *Stringhe*
- *Numeri*
- *Booleani*

*Reference:*

- *Oggetti*
- *Array*
- *Funzioni*

# COPY VS REFERENCE

## Esempio: Copy

```
let persona = 'Peter Parker'  
let superhero = persona  
  
superhero = 'Spiderman'  
console.log(persona)    // risultato: 'Peter Parker'  
console.log(superhero) // risultato: 'Spiderman'
```

## Esempio: Reference

```
let persona = {  
  nome: 'Peter Parker'  
}  
  
let superhero = persona  
superhero.nome = 'Spiderman'  
  
console.log(persona.nome)    // risultato: 'Spiderman'  
console.log(superhero.nome) // risultato: 'Spiderman'
```



# STRUTTURE DI CONTROLLO

---

*Funzioni come first class citizen*

# FUNZIONI COME FIRST CLASS CITIZEN

---

*Caratteristica di tutte le funzioni in javascript*

*Significa che le funzioni in javascript si comportano come valori*

*Permette di:*

- *Assegnare una funzione ad una variabile*
- *Assegnare una funzione come valore in una struttura dati*
- *Passare funzione come parametro ad un'altra funzione*
- *Usare una funzione come valore di ritorno*

# FUNZIONI COME FIRST CLASS CITIZEN

```
function logToconsole(){ return 'hello world!' }

// Assegnazione ad una variabile
let myFunction = logToconsole
let output = myFunction()
console.log(output)

// Assegnazione ad una proprietà di un oggetto
let persona = {
  name: 'pippo',
  hello: logToconsole
}
let out = persona.hello()
console.log(out)
```

**ATTENZIONE:** Non stiamo assegnando il valore di ritorno ma la funzione!!

# FUNZIONI COME PARAMETRI

*Le funzioni in Javascript si comportano come valori*

```
// Dichiarazione
function logToConsole(){
    console.log('interval')
    console.log('2 secondi')
}

// Utilizzo
setTimeout(logToConsole, 2000)
```

# FUNZIONI ANONIME

*Si comportano solo come valori*

```
let logFullName = function(firstName, lastName){
  console.log(firstName + ' ' + lastName)
}
logFullName()
```

*E possibile assegnare una funzione ad una proprietà di un oggetto*

```
// funzioni anonime
let persona = {
  firstName: 'Peter',
  lastName: 'Parker',
  fullname: function (){
    return this.firstName + ' ' + this.lastName
  }
}
let fullname = persona.fullname()
console.log(fullname)
```

# ESEMPI - FUNZIONE ANONIMA COME PARAMETRO

*Esegue la funzione ogni 2 secondi*

```
// SINTASSI: setInterval(funzione, intervallo)
setInterval(function(){
    console.log('interval')
    console.log('2 secondi')
}, 2000)
```

*Esegue la funzione 1 sola volta dopo 1 secondo*

```
// SINTASSI: setTimeout(funzione, ritardo)
setTimeout(function(){
    console.log('Ritardo di 5 secondi')
}, 5000)
```

# DOCUMENT OBJECT MODEL (DOM)

A knight in dark, ornate armor stands before an open book. The book's pages are a vibrant, glowing blue and white, emitting a bright light that illuminates the surrounding environment. The knight's armor is detailed with gold accents and a chainmail hauberk. The background is a soft-focus landscape with green trees and a pale sky.

# DOCUMENT OBJECT MODEL (DOM)

---

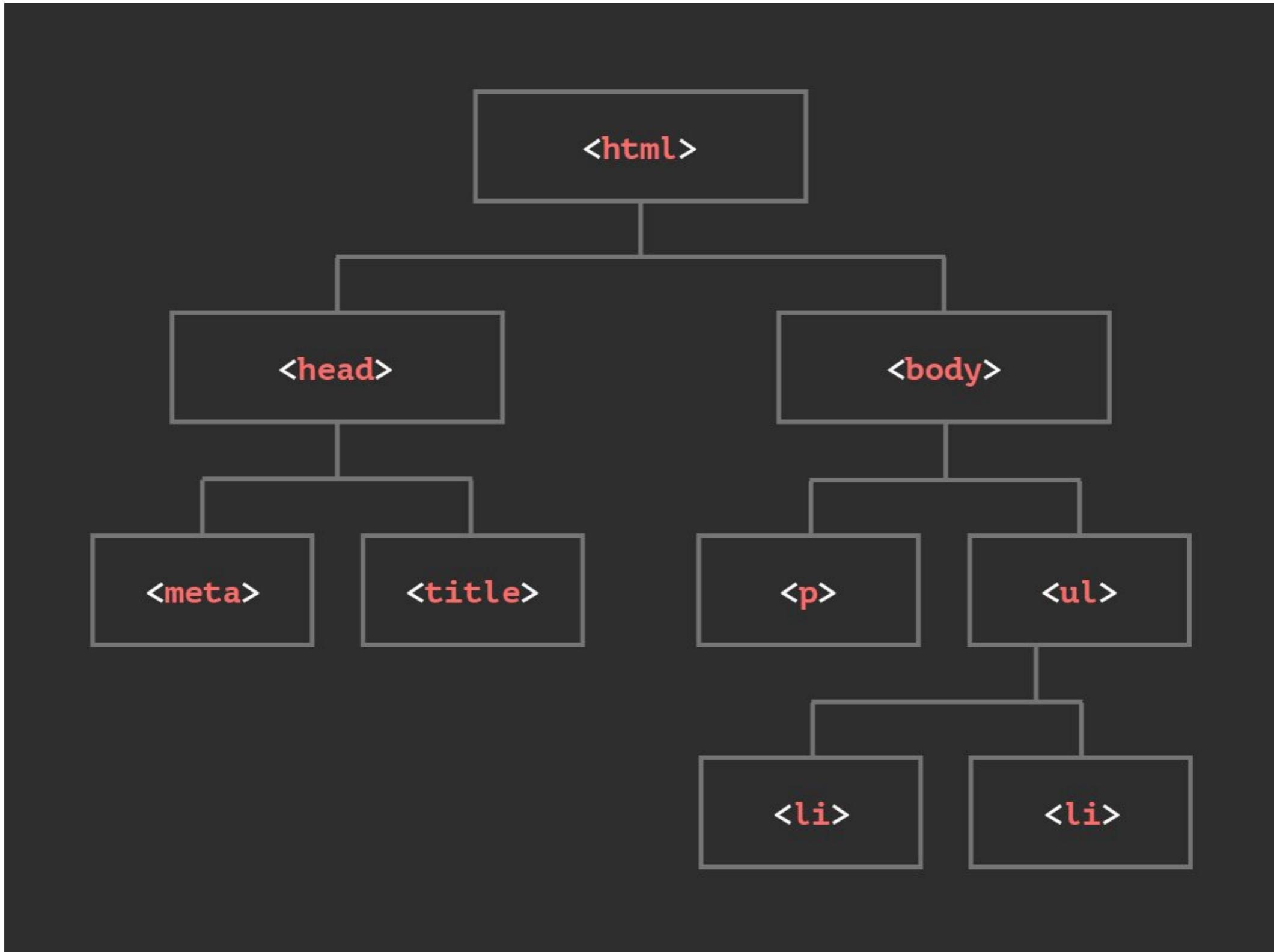
*Controllo degli elementi visivi della pagina*

*E una struttura dati accessibile tramite la variabile globale “document”*

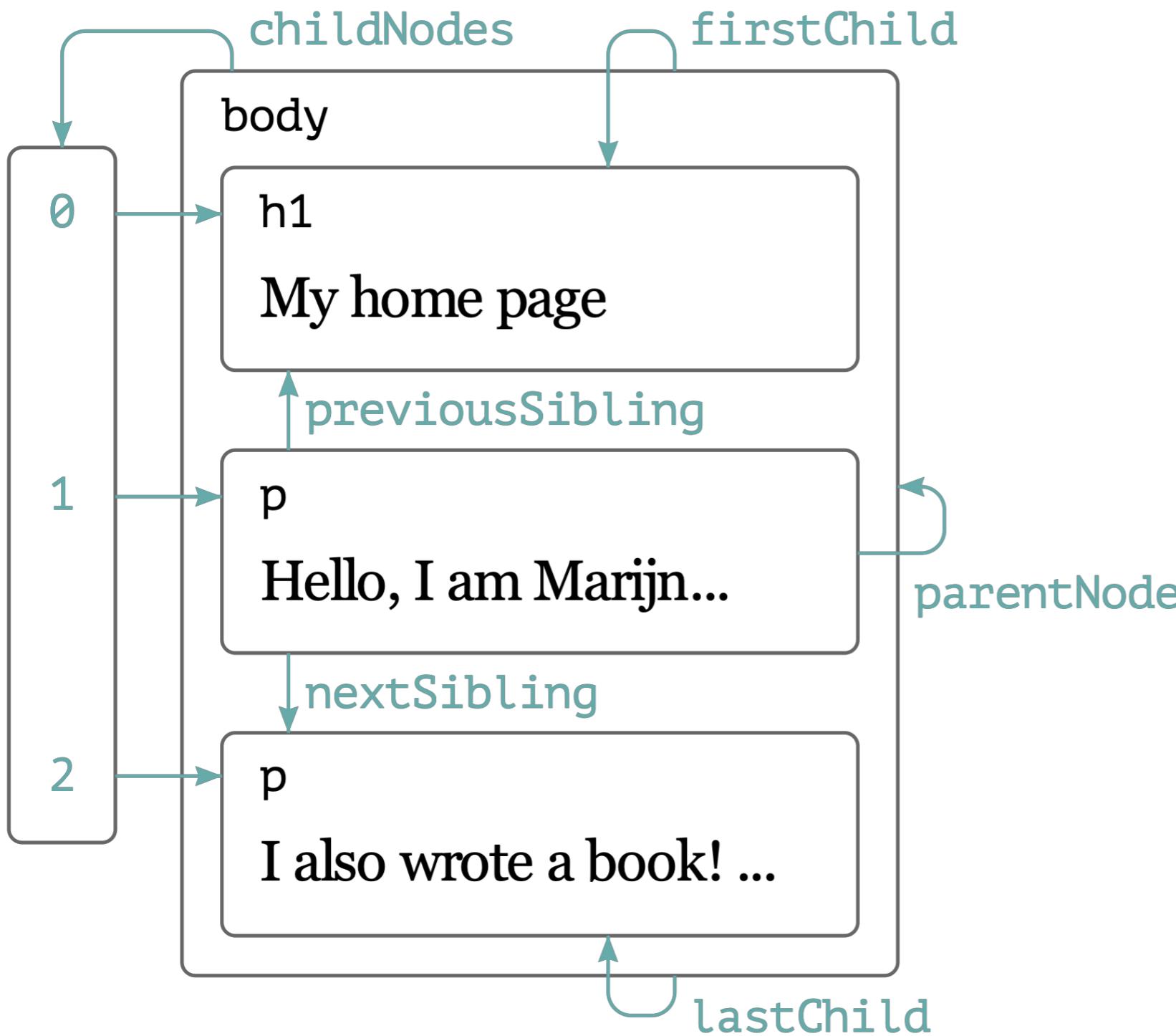
*Permette di:*

- *Cercare elementi*
- *Modificare elementi (ES: proprietà e stili)*
- *Eliminare elementi*
- *Creare elementi*
- *Eseguire codice in risposta ad eventi (ES: azioni dell'utente, eventi del browser)*
- *Accedere alle API del browser (ES: Leggere l'url della pagina, ottenere la posizione dell'utente)*

# DOM - RELAZIONI - PROPRIETÀ CHILDREN



# DOM - RELAZIONI



Ogni oggetto ha:

- Padre (Parent)
- Siblings (Vicini)
- Children (figli)

# DOM - RICERCA DELLA REFERENCE

*Restituisce un oggetto (oppure undefined se non trova niente)*

```
// cerca in base all id del tag html ( es: <div id="myID"></div> )
document.getElementById('#myID')

// cerca in base ad un selettore (identico al selettore css)
document.querySelector('div#myID')
```

*Restituisce un array di oggetti (oppure array vuoto se non trova niente)*

```
// cerca in base ad una classe tag html ( es: <div class="myClass"></div> )
document.getElementsByClassName('myClass')

// cerca in base al tag html ( es: <h1></h1> )
document.getElementsByTagName('h1')

// cerca in base ad una classe tag html ( es: <h1 class="myClass"></h1> )
document.querySelectorAll('h1.myClass')
```

# ESEMPI - DOM - MODIFICA ELEMENTO

```
// Cerco la Referenza
let reference = document.getElementById('#myID')

// Controllo che la referenza esita e modifico l'oggetto
if(reference){
    // Generico
    reference.style.color = '#fff'
    reference.style.backgroundColor = '#4e4e4e'
    reference.style.padding = '10px'

    // Paragrafo
    reference.innerHTML = 'Nuovo testo del paragrafo'

    // Ancora
    reference.href = 'https://google.com'

    // Immagine
    reference.src = "images/gattini.jpg"
}
```

# ESEMPI - DOM - AGGIUNTA NUOVO ELEMENTO

## *Creazione dell'elemento*

```
// Creazione Elemento
var paragrafo = document.createElement('p')

// Modifica dell elemento
paragrafo.innerText = "Sono un nuovo paragrafo"
paragrafo.style.color = 'blue'
```

## *Inserimento nella pagina*

```
// Inserimento nel dom
var container = document.getElementById('container')
container.append(paragrafo)
```

# ESEMPI - DOM - AGGIUNTA NUOVO ELEMENTO

## *Spostamento*

```
var container = document.getElementById('container')
container.prepend(paragrafo)
```

## *Duplicazione*

```
var container = document.getElementById('container')
let clone = paragrafo.cloneNode()
container.append(clone)
```

**ATTENZIONE:** *cloneNode* clona solo l'elemento non i figli

*Se si vuole clonare tutti i discendenti usare: *cloneNode(true)**

# ESEMPI - DOM - ELIMINAZIONE ELEMENTO

---

```
// Tramite l'elemento stesso
let pippo = document.getElementById('pippo')
pippo.remove()
```

```
// Tramite il parent
let container = document.getElementById('container')
container.removeChild(pippo)
```

# EVENTI - DOM

A vibrant night concert scene. In the foreground, the silhouettes of a large crowd of people are visible, many with their hands raised. The stage is bathed in a variety of bright, colorful lights, including red, green, blue, and yellow beams that create a dynamic and energetic atmosphere. The background is dark, making the stage lights stand out.

# EVENTI DOM

---

*Avvengono su un elemento del DOM*

- *In seguito ad azioni svolte dall'utente su quell'elemento (ES: Click, Submit ...)*
- *In seguito al cambio di stato di elementi del DOM (ES: Load, Resize)*
- *In maniera programmatica*

*Possono essere:*

- *Eventi Nativi del browser*
- *Eventi custom creati dallo sviluppatore*

# ESEMPI - DOM - EVENTI MOUSE

```
// Referenza di un paragrafo
let pippo = document.getElementById('pippo')

// Aggiungo l'handler
pippo.addEventListener('click', function(event){
    console.log('hai cliccato il paragrafo')
})
```

## ALTRI EVENTI DEL MOUSE

- *dblclick* : doppio click
- *mouseover*: mouse sopra l'elemento
- *mouseout*: mouse non più sopra elemento

# ESEMPI - DOM - EVENTI FORM/INPUT

```
// Referenza di un input
let firstName = document.getElementById('firstName')

// Aggiungo l'handler
firstName.addEventListener('input', function(event){
    console.log('hai cliccato il paragrafo')
})
```

## ALTRI EVENTI DEGLI INPUT

- *focus*: input attivo
- *change*: cambio del valore (radiobutton e selectbox)

## ALTRI EVENTI DEI FORM

- *submit*: submit del form
- *reset*: reset del form

# ESEMPI - DOM - EVENTI TASTIERA

```
// Eseguito quando si preme un tasto corrispondente a lettere  
numeri o simboli  
document.addEventListener('keypress', function(event){  
    console.log(event.key)  
})
```

## ALTRI EVENTI TASTIERA

- *keydown*: pressione di qualsiasi tasto
- *keyup*: rilascio di qualsiasi tasto

# ESEMPI - DOM - EVENTI DOCUMENT

```
// Eseguito quando la pagina e tutti gli script sono stati  
// caricati  
document.addEventListener('DOMContentLoaded', function(event){  
    console.log(event.key)  
})
```

## ALTRI EVENTI GENERALI

- **scroll**: scroll della pagina
- **resize**: resize della finestra

# ESEMPI - DOM - PREVENT

```
// Referenza di un paragrafo
let submit = document.getElementById('submit')

// Aggiungo l'handler
submit.addEventListener('click', function(event){
    // impedisce al click di fare submit del form
    event.preventDefault()
})
```

**ATTENZIONE:** non è possibile ripristinare l'evento.

Si può aggirare usando una condizione:

```
let disabled = true
submit.addEventListener('click', function(event){
    if(disabled){ event.preventDefault() }
})
```

# ESEMPI - DOM - TRIGGER PROGRAMMATICO

```
// Referenza di un bottone submit in un form
let submit = document.getElementById('submit')
// Esegue il click in maniera programmatica
submit.click()
```

```
// Referenza di un form
let form = document.getElementById('myForm')
// Esegue il submit in maniera programmatica
form.submit()
```

# ESEMPI - DOM - EVENTI CUSTOM

```
// Creazione Evento custom
const event = new CustomEvent(
  'pippoEvent',
  { detail: { 'Hai studiato?': true } }
)

// Trigger evento custom
document.dispatchEvent(event)
```

# EVENT BUBBLING

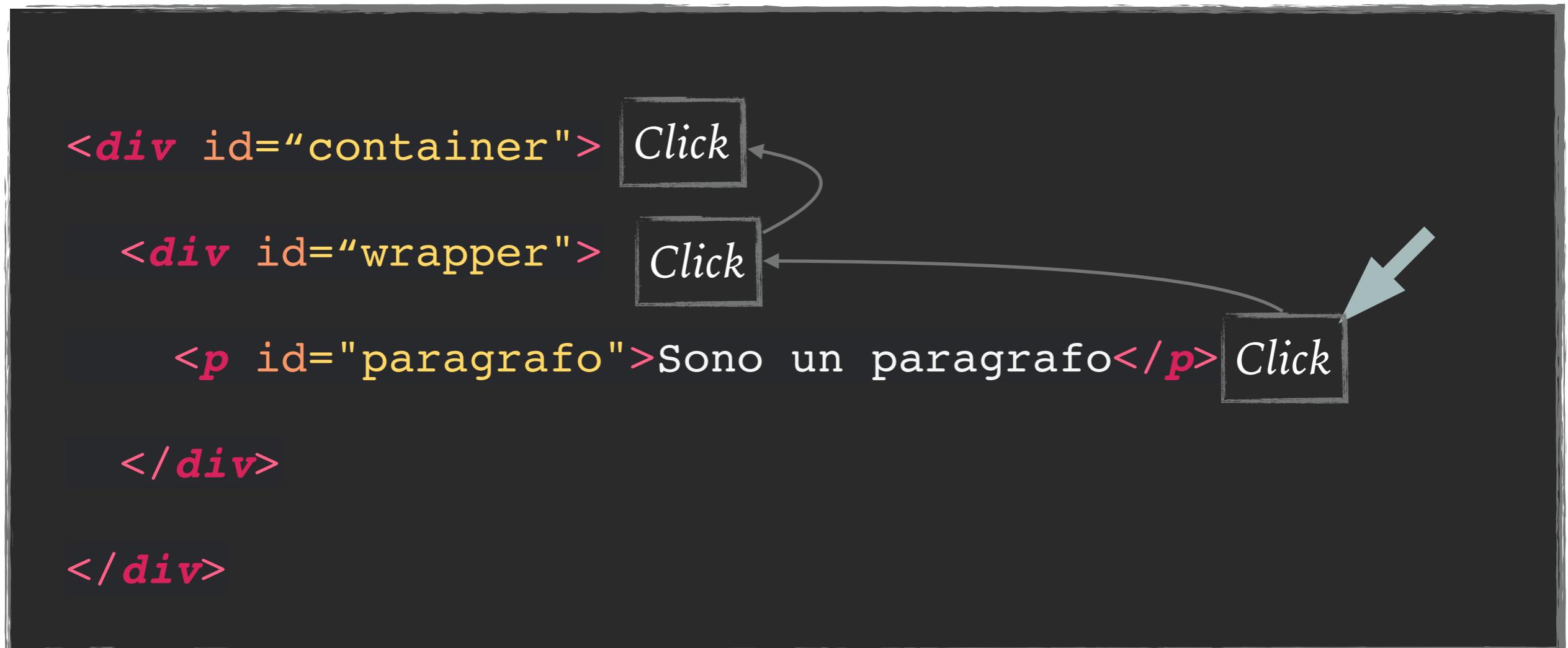
A photograph of a young girl with dark hair, smiling broadly with her mouth open. She is underwater, surrounded by numerous small, white bubbles. The background is a clear blue water. The image is framed by a thick white border.

# EVENT BUBBLING

*Alcuni tipi di evento si propagano automaticamente verso l'alto*

*L'evento si scatena allo stesso modo su tutti gli antenati*

*L'ordine segue la struttura del dom dal basso verso l'alto*



# ESEMPI - DOM - BUBBLING

## HTML

```
<div id="container">
  <div id="wrapper">
    <p id="paragrafo">Sono un paragrafo</p>
  </div>
</div>
```

## Javascript

```
let container = document.getElementById('container')

container.addEventListener('click', function(event){
  // Elemento da cui origina l'evento (Elemento su cui si ha cliccato)
  console.log(event.target)

  // Elemento su cui è attaccato l'handler (container)
  console.log(event.currentTarget)
})
```

# ESEMPI - DOM - FERMARE IL BUBBLING

HTML

```
<div id="container">
  <div id="wrapper">
    <p id="paragrafo">Sono un paragrafo</p>
  </div>
</div>
```

Javascript

```
let container = document.getElementById('container')
let wrapper = document.getElementById('wrapper')

wrapper.addEventListener('click', function(event){
  // Fermare il bubbling su questo elemento
  event.stopPropagation()
})
```

# BUONE PRATICHE - DOM



**Don't cross the streams, it would be bad!**

# DOM - BEST PRACTICES

Male!

```
let ul = document.createElement('ul')
body.append(ul)

for(let el of list){
  let el = document.createElement('li')
  li.innerText = el.name
  ul.append(li)
}
```

Bene!

```
let ul = document.createElement('ul')
for(let el of list){
  let el = document.createElement('li')
  li.innerText = el.name
  ul.append(li)
}

body.append(ul)
```

# DOM - BEST PRACTICES

Male!

```
let div = document.createElement('p')
let img = document.createElement('img')
let a = document.createElement('a')

body.append(div, img, a)
```

Bene!

```
let div = document.createElement('p')
let img = document.createElement('img')
let a = document.createElement('a')

let fragment = document.createDocumentFragment()
fragment.append(div, img, a)

body.append(fragment)
```

# DOM - BEST PRACTICES

Male!

```
let pippo = document.getElementById('pippo')

pippo.style.color = 'gray'
pippo.style.fontSize = '18px'
pippo.style.textAlign = 'center'
pippo.style.TextAlign = 'center'
pippo.style.fontStyle = 'italic'
pippo.style.borderBottom = 'solid 1px gray'
pippo.style.padding = '5px'
```

Bene!

```
let pippo = document.getElementById('pippo')

// Utilizzo una classe e applico lo stile nel CSS
pippo.classList.add('pippo-style')
```

A large collection of colorful brooms and brushes hanging on a wall. The brushes have various colored bristles including red, green, blue, purple, black, and white. They are mounted on a light-colored wooden board.

**SCOPE**

# SCOPE

OUT -> IN

```
let nome = 'Peter Parker'

function myFunction(){
  console.log(nome)
}

myFunction()
// CONSOLE LOG: 'Peter Parker'
```

IN -> OUT

```
let nome = 'Peter Parker'
function myFunction(){
  nome = 'Tony Stark'
  console.log('Dentro funzione: ', nome)
}

myFunction()
console.log('Fuori funzione: ', nome)
// CONSOLE LOG: Dentro funzione: Tony Stark
// CONSOLE LOG: Fuori funzione: Tony Stark
```

# SCOPE

## Shadowing

```
let nome = 'Peter Parker'

function myFunction(){
  let nome = 'Tony Stark'
  console.log('Dentro funzione: ', nome)
}

myFunction()
console.log('Fuori funzione: ', nome)
// CONSOLE LOG: Dentro funzione: Tony Stark
// CONSOLE LOG: Fuori funzione: Peter Parker
```

# SCOPE - LET VS VAR

OUT -> IN

```
let lista = [ 'Peter Parker', 'Tony Stark' ]
for(let person of lista){
    // codice
}

console.log(person)
// ERROR: Uncaught ReferenceError: person is not defined
```

IN -> OUT

```
let lista = [ 'Peter Parker', 'Tony Stark' ]
for(var person of lista){
    // codice
}

console.log(person)
// CONSOLE LOG: Tony Stark
```

# THIS - DYNAMIC SCOPE

```
let person = {
  name: 'Peter Parker',
  logName: function() {
    setTimeout(function() {
      console.log('NAME', this.name)
      console.log('THIS', this)
    }, 1000)
  }
}

person.logName()
// CONSOLE LOG: NAME
// CONSOLE LOG: THIS Window{window: Window, self: Window,
document: document, name: '', location: Location...}
```

# SCOPE - HOISTING

## LET

```
let nome = 'Peter Parker'  
function myFunction(){  
    console.log(nome)  
    let nome = 'Tony Stark'  
}  
myFunction()  
// ERRORE: Uncaught ReferenceError: Cannot access 'nome' before initialization
```

## VAR

```
let nome = 'Peter Parker'  
function myFunction(){  
    console.log(nome)  
    var nome = 'Tony Stark'  
}  
myFunction()  
// CONSOLE LOG: undefined
```



# ARROW FUNCTIONS

# ARROW FUNCTION - SINTASSI

```
let funzione = ( parametro1, parametro2) => {  
    console.log(parametro1+ ' '+parametro2)  
}
```

Parametri

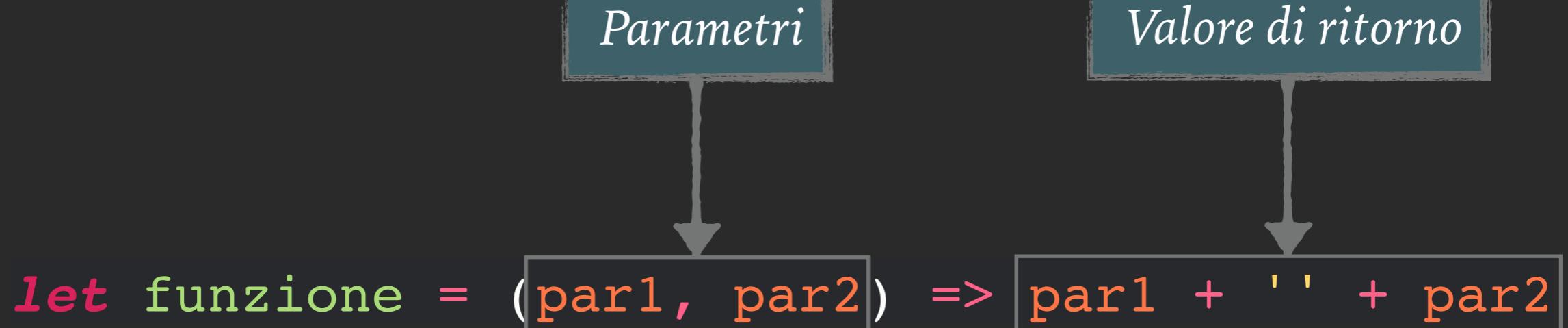
Blocco di codice

# THIS - LEXICAL SCOPE

```
let person = {
    name: 'Peter Parker',
    logName: function(){
        setTimeout(()=>{
            console.log('NAME', this.name)
            console.log('THIS', this)
        }, 1000)
    }
}

person.logName()
// CONSOLE LOG: NAME 'Peter Parker'
// CONSOLE LOG: THIS {name: 'Peter Parker', logName: f}
```

# ARROW FUNCTION - SINTASSI BREVE



*Nel caso ci sia 1 SOLO parametro si possono omettere anche le parentesi tonde*

```
let funzione = parametro1 => parametro1 * 2
```

A black silhouette of two people playing baseball on a field. One person on the left is in a batting stance, and another person on the right is in a pitching stance. A baseball is shown in the air between them. The background shows a blue sky with a few clouds and a dark silhouette of trees at the bottom.

**REQUEST**

# REQUEST

---

*Comunicazione con una risorsa remota*

- *Ottenerne dati*
- *Inviare dati (ES: registrazione utente, update profilo)*
- *Eseguire azioni (ES: eliminare un post, confermare un ordine)*

*Si possono eseguire request:*

- *Tramite form (HTML)*
- *In Background tramite AJAX (Javascript)*
- *Tra 2 server (PHP, Java, Python, Node ....)*

# REQUEST - DATI



*REQUEST*

*DATA*

*RESPONSE*



*REQUEST*

*DATA*  
*RESPONSE*



*REQUEST*

*DATA*

*DATA*  
*RESPONSE*



# REQUEST - METADATI



## ▼ Response Headers

[View source](#) **META-DATA**

```
allow: OPTIONS, GET
content-length: 7711
content-type: application/json
date: Wed, 09 Feb 2022 14:23:35 GMT
server: nginx/1.21.1
vary: Accept, Origin
x-frame-options: SAMEORIGIN
```

[View source](#)

## ▼ Request Headers

**META-DATA** [View source](#)

```
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en;q=0.9,it;q=0.8,en-US;q=0.7
authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl9Tk2ZDY3NzU5NiIsInVzZXJfaWQiOjIsInJvbGUiOiJhZG1pbjIsInVzZXJuYW1lIjoicjIiwiY3JlYXRlZCI6bnVsbH0.80uhBNyw2xoCgJX0q9Z99Xq1uUZPuxg7mUUxFu_ZeIA4
Connection: keep-alive
```

# REQUEST - FETCH

```
fetch('https://reqres.in/api/users')
  .then((response)=>{
    // Qui Posso esaminare:
    // - Stato della risposta
    // - Metadati (ES: headers)
    console.log(response)
    return response.text()
  })
  .then((data)=>{
    // In questo blocco ho accesso ai dati
    console.log(data)
  })
}
```

# REQUEST - RESPONSE CODES

---

## **SUCCESS (2XX)**

*200 = OK*

*202 = Created*

*204 = No Content*

## **REDIRECTION (3XX)**

*301 = Moved Permanently*

*307 = Temporary Redirect*

## **CLIENT ERROR (4XX)**

*400 = Bad Request*

*401 = Unauthorized*

*403 = Forbidden*

*404 = Not Found*

## **SERVER ERROR (5XX)**

*500 = Internal server error*

*502 = Bad Gateway*

# REQUEST - FETCH - ERROR HANDLING

```
fetch('https://reqres.in/api/users')
  .then((response)=>{
    if(response.status<205){ return r.json() }
    throw Error(response.statusText)
  })
  .then((data)=>{
    console.log(data)
  })
  .catch((error)=>{
    console.log('Errore ', error)
  })
}
```

# REQUEST - FETCH - FINALLY

```
fetch('https://reqres.in/api/users')
  .then((response)=>{
    if(response.status<205){ return r.json() }
    throw Error(response.statusText)
  })
  .then((data)=>{
    console.log(data)
  })
  .catch((error)=>{
    console.log('Errore ', error)
  })
  .finally(()=>{
    console.log('END')
  })
}
```

*FINALLY verrà eseguito sia in caso di SUCCESSO sia in caso di ERRORE*

*FINALLY non passa nessun parametro alla funzione*



# REST

# REST

---

*Quale l'url giusto per creare un nuovo Utente?*

- A) *HTTPS://MYSITE.COM/API/ADD-USER*
- B) *HTTPS://MYSITE.COM/API/USER/CREATE*
- C) *HTTPS://MYSITE.COM/API/CREATE/USER*
- D) *HTTPS://MYSITE.COM/API/USER?action=create*
- E) *HTTPS://MYSITE.COM/API/USER/NEW*

# REST

---

## *REST*

- *Separazione/Indipendenza tra server e client*
- *Comunicazione tramite HTTP*
- *Stateless*
- *Cacheabile*
- *Interfaccia uniforme*

## *Interfaccia uniforme:*

- *Ogni risorsa deve avere 1 solo URL/baseURL*
- *Utilizzo dei HTTP Verbs/Methods per definire il tipo di operazione*
- *Utilizzo di queryParams per attività di filtraggio/Ricerca dei risultati*
- *Struttura degli URL consistente e prevedibile*

`HTTPS://MYSITE.COM/API/USER`

`GET`

*Lista degli utenti*

`HTTPS://MYSITE.COM/API/USER`

`POST`

*Crea un nuovo utente*

`HTTPS://MYSITE.COM/API/USER/42`

`GET`

*Dettaglio User con ID=42*

`HTTPS://MYSITE.COM/API/USER/42`

`PUT`

*Modifica User con ID=42*

`HTTPS://MYSITE.COM/API/USER/42`

`DELETE`

*Elimina User con ID=42*

`HTTPS://MYSITE.COM/API/USER/42/ORDER`

`GET`

*Lista Order dell User con ID=42*

`HTTPS://MYSITE.COM/API/USER/42/ORDER`

`POST`

*Crea nuovo Order dell User con ID=42*

`HTTPS://MYSITE.COM/API/ORDER/12`

`GET`

*Dettaglio ordine con ID=12*

# REQUEST - FETCH - OPTIONS

```
fetch(  
  'https://reqres.in/api/users',  
  {  
    method: "POST",  
    headers: {  
      'Accept': 'application/json',  
      'Content-Type': 'application/json'  
    },  
    body: JSON.stringify({a: 1, b: 2})  
  }  
)  
.then((response)=>response.json())  
.then((data)=>console.log(data))
```

[https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API/Using\\_Fetch](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)

A woman with long brown hair, wearing a white tank top, is smiling down at a young girl. They are both under a white sheet or blanket. The woman is holding the girl's small hand in her larger one. The scene is softly lit, creating a warm and intimate atmosphere.

PROMISE

# ESEMPI - FETCH - PROMISE

```
let request = fetch('https://reqres.in/api/users')
console.log(request)

request
  .then((response)=>{
    console.log(response.status)
    return response.json()
  })
  .then((data)=>{ console.log(data) })
```

▼ **Promise {<pending>}** ⓘ  
▼ [[Prototype]]: Promise  
► **catch: f catch()**  
► **constructor: f Promise()**  
► **finally: f finally()**  
► **then: f then()**

CONSOLE

# ESEMPI - CUSTOM PROMISE

```
let prom = new Promise(  
  (resolve, reject)=>{  
    setTimeout(()=>{  
      if(Math.random() < 0.5) resolve('OK')  
      else reject('non OK')  
    }, 2000)  
  }  
) ;  
  
prom.then(  
  (results)=>{ console.log('Results', results) } ,  
  (error)=>{ console.log('Error', error) }  
)  
.catch((error)=>{  
  console.log('Error', error)  
})  
.finally(()=>{  
  console.log('END')  
})
```

**ATTENZIONE:** SE si passa una seconda funzione nel blocco THEN andrà a sostituire il blocco CATCH che non verrà eseguito.



# BROWSER APIs

# BROWSER API - GEOLOCATION

```
navigator.geolocation.getCurrentPosition(  
  (position)=>{  
    console.log(position)  
  },  
  (error)=>{  
    console.log(error)  
}  
)
```

▼ *GeolocationPosition*

  ▼ *coords*: GeolocationCoordinates

- accuracy: 13.784
- altitude: null
- altitudeAccuracy: null
- heading: null
- latitude: 45.0751747
- longitude: 7.6006638
- speed: null

SUCCESS

▼ *GeolocationPositionError*

  code: 1

  message: "User denied Geolocation"

► [[Prototype]]: GeolocationPositionError

*navigator.geolocation.watchPosition*

Ha la stessa sintassi ma esegue la funzione ogni volta che la posizione cambia

# BROWSER API - LOCALSTORAGE

*Storage locale del browser Chiave-valore. I valori sono SEMPRE string o null*

```
// SCRITTURA  
window.localStorage.setItem('userName', 'pippo')  
  
// LETTURA  
let name = window.localStorage.getItem('userName')  
// string || null  
  
// RIMOZIONE di una chiave  
window.localStorage.removeItem('userName')  
  
// RIMOZIONE di tutto  
window.localStorage.clear()
```

*window.sessionStorage*

*Ha la stessa sintassi ma i dati esistono solo nel tab corrente.*

*Tutti i dati vengono eliminati alla chiusura del tab o del browser*

# ESEMPI - LOCALSTORAGE - JSON

```
// SCRITTURA
let persona = {
  firstName: 'paperino',
  lastName: 'paolino',
}
let personaJSON = JSON.stringify(persona)
window.localStorage.setItem('user', personaJSON)

// LETTURA
let personaJSON = window.localStorage.getItem('user')
let persona = JSON.parse(personaJSON)
```